UNIVERSITÄT BONN

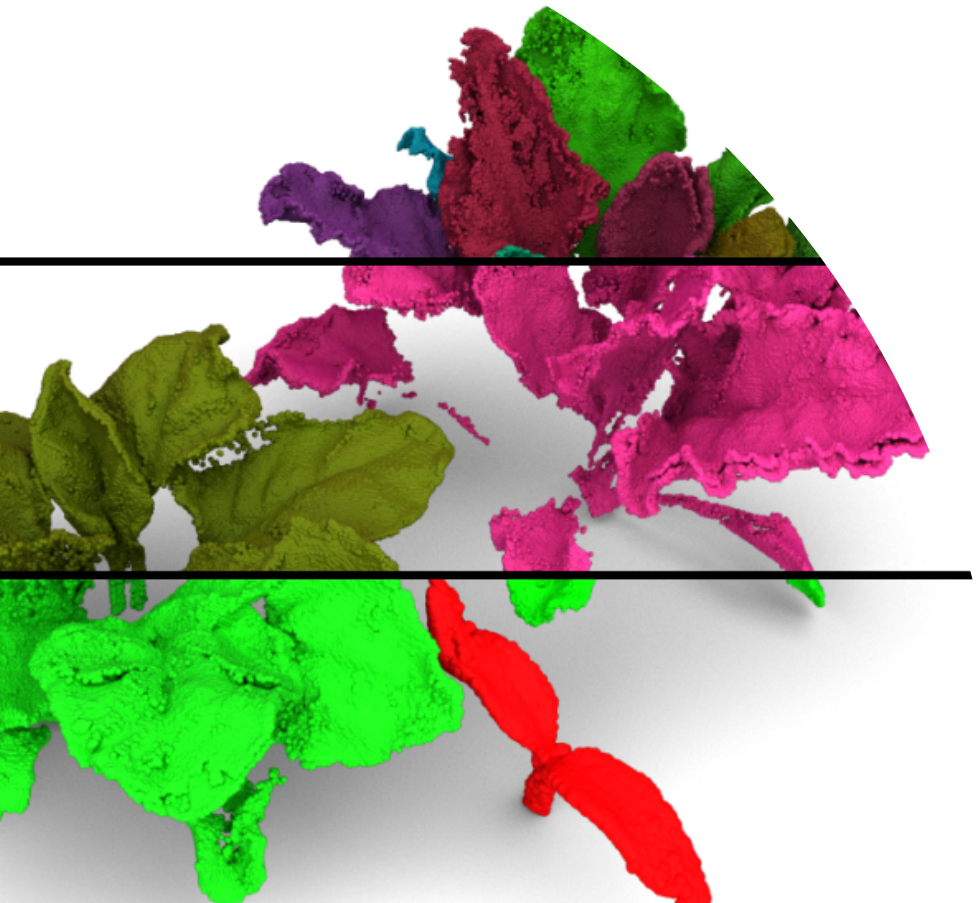
Inaugural-Dissertation
zur Erlangung des Grades
Doktor der Ingenieurwissenschaften (Dr.-Ing.)
Agrar-, Ernährungs- und Ingenieurwissenschaftliche Fakultät
der Rheinischen Friedrich-Wilhelms-Universität Bonn
Institut für Geodäsie und Geoinformation

Unsupervised Learning for In-Field Phenotyping Leveraging Domain Knowledge

von

Gianmarco Roggiolani

aus
Rom, Italien



Bonn 2026

Referent:
Prof. Dr. Cyrill Stachniss, University of Bonn, Germany

1. Korreferent:
Prof. Dr. Uwe Rascher, University of Bonn, Germany

Tag der mündlichen Prüfung: 20.01.2026

Angefertigt mit Genehmigung der Agrar-, Ernährungs- und Ingenieurwissenschaftlichen Fakultät der Universität Bonn

Zusammenfassung

Die zunehmende Weltbevölkerung und die unzureichende Nachhaltigkeit herkömmlicher Praktiken stellen erhebliche Herausforderungen für unser landwirtschaftliches System. Um die steigende Nachfrage zu decken und Ressourcen zu schonen, müssen Erträge gesteigert und gleichzeitig nachhaltige Methoden entwickelt werden.

Robotersysteme stellen eine potentiell nachhaltigere Alternative zu traditionellen landwirtschaftlichen Verfahren. Durch gezieltes Jäten statt flächendeckender Besprühung können sie den Einsatz von Agrochemikalien deutlich reduzieren. Zudem überwachen sie kontinuierlich den Pflanzenzustand und liefern wertvolle Daten, die Züchter und Agronomen zur Entwicklung widerstandsfähigerer und ertragreicherer Sorten nutzen können.

Roboter sind dabei auf leistungsfähige Perzeptionssysteme angewiesen, die präzise Messungen im Feld ermöglichen. Diese Perzeptionssysteme nutzen oft datengetriebene Ansätze, die durch das Lernen aus manuell erstellten Annotationen von Daten charakterisiert sind. Zur adäquaten Wahrnehmung ihrer Umgebung ist es für die Wahrnehmungssysteme erforderlich, auf umfangreiche Mengen annotierter Daten zuzugreifen, die eine Vielzahl von Szenarien abdecken. Dazu gehören unterschiedliche Pflanzenwachstumsraten, Lichtverhältnisse, Bodenbeschaffenheiten sowie verschiedene Arten von Nutzpflanzen. Die hohen Kosten sowie der erhebliche Zeitaufwand, die mit der Erstellung annotierter Daten verbunden sind, stellen ein echtes Problem dar, welches den Einsatz von Robotersystemen maßgeblich behindert. Um die Abhängigkeit von manuell annotierten Daten zu reduzieren, können wir Techniken entwickeln, die Vorwissen über Feldaufbau und Pflanzeneigenschaften nutzen, um datengetriebene Ansätze zu trainieren.

Der wesentliche Beitrag dieser Arbeit liegt in der Entwicklung innovativer Wahrnehmungstechniken, die darauf abzielen, das Szenenverständnis durch Robotersysteme zu optimieren. Dabei liegt ein besonderer Fokus auf der Minimierung des Bedarfs an manuell annotierten Daten. In dieser Arbeit stellen wir einen methodischen Ansatz zur Identifizierung von Unkraut, Nutzpflanzen, Einzelpflan-

zen sowie Einzelblättern vor, der auf manuell annotierten Daten basiert. Wir zeigen dann, wie Kenntnisse über die landwirtschaftliche Umgebung effektiv eingesetzt werden können, um die Effizienz ohne zusätzliche annotierte Daten zu erhöhen. Danach stellen wir einen methodischen Ansatz zur Differenzierung zwischen Nutzpflanzen und Unkraut sowie zur Identifizierung einzelner Pflanzen auf den Feldern, wobei wir auf die Verwendung von Annotationen verzichten. Wir werden darlegen, auf welche Weise die Segmentierung von einzelnen Blättern in drei Dimensionen durch die gezielte Berücksichtigung der Pflanzenstruktur optimiert werden kann. Darauf präsentieren wir unseren methodischen Ansatz zur Generierung realistischer 3D Blätter mit definierten Längen und Breiten, um die Leistungsfähigkeit der bestehenden Verfahren zur Schätzung diverser Merkmale signifikant zu verbessern.

Insgesamt leistet diese Dissertationsschrift einen wesentlichen Beitrag zur Analyse landwirtschaftlicher Daten, indem sie verschiedene Aufgaben adressiert, die vom semantischen Verständnis von Nutzpflanzen und Unkräutern bis zur präzisen Bestimmung von Blattmerkmalen, wie der Breite und Länge der Blattspreite, reichen. Die in dieser Arbeit präsentierten Ansätze der Bildverarbeitung tragen dazu bei, die Identifizierung und Messung von Nutzpflanzen, einzelnen Pflanzen sowie einzelnen Blättern zu verbessern, während gleichzeitig der Bedarf an manuell beschrifteten Daten signifikant reduziert wird. Wir nutzen dabei vorhandenes Vorwissen, um die Effizienz bestehender Technologien zu optimieren und automatisch annotierte Daten zu generieren, die für datengetriebene Ansätze von Bedeutung sind. Durch unsere Maßnahmen verringern wir sowohl den Aufwand als auch die Zeit, die für die Annotation eines Datensatzes zum semantischen Verständnis erforderlich sind, auf weniger als die Hälfte. Dies stellt einen signifikanten Fortschritt in Richtung eines effizienteren und robusteren Wahrnehmungssystems für landwirtschaftliche Anwendungen dar.

Abstract

The growing world population and the unsustainability of common farming practices are challenging our agricultural production system, which has to cope with the increased demand for food, feed, fuel, and fiber, without draining the natural resources, worsening climate change, or compromising environmental biodiversity. We need to rethink our whole farming system to increase the yield per area unit and improve the sustainability of our methods.

Robotic systems have the potential to offer a more sustainable alternative to standard practices. They can perform targeted weeding instead of uniformly spraying the whole field, thus reducing the use of agrochemicals. Robots can also continuously monitor the state of plants in the field, providing measurements that breeders and agronomists can use to develop more resilient and high-throughput crop varieties.

Robots need robust perception systems to provide accurate in-field measurements. Such perception systems are usually data-driven approaches learning from manually produced examples, also called labeled data. To correctly understand their surroundings, the perception systems need access to vast amounts of labeled data, covering all possible scenarios, i.e., different plant growths, light conditions, soil textures, and crop species. The high cost and time required to produce labeled data are the bottlenecks that limit the adoption of robotic systems.

However, in the agricultural domain, we can exploit prior knowledge about the fields' arrangements and the plants' characteristics to enhance the abilities of the perception systems, while simultaneously reducing the need for labeled data for data-driven approaches.

The main contribution of this thesis is a set of novel perception techniques to improve the scene understanding of robotic systems with a focus on reducing the requirements for manually annotated data. First, we present an approach to identify weeds, crops, single plants, and single leaves using manually annotated data. Then, we show how to exploit the knowledge about the agricultural environment to boost the performance of all tasks without additional annotated data. Our third contribution is an approach to distinguish crops from weeds, and our

fourth contribution is an approach to identify single plants in the fields. Both of them do not require annotated data. We then present how to improve single-leaf segmentation in 3D exploiting the plant structure as the fifth contribution. Finally, we present our approach to generate realistic 3D leaves of known lengths and widths to enhance the capabilities of existing trait estimation approaches.

In summary, this thesis contributes to the interpretation of agricultural data for different tasks, from the semantic understanding of crops and weeds to the estimation of leaf traits, such as the width and length of the leaf blade. The computer vision approaches presented in this thesis allow for more accurate identification and measurement of crops, single plants, and single leaves with reduced requirements for manually labeled data. We exploit the prior knowledge about the agricultural domain to boost the performance of existing techniques and to produce automatically annotated data to be used in data-driven approaches. We cut to less than half the cost and time required to annotate a dataset for semantic understanding, thus making a concrete step toward a more efficient and robust perception system for farming tasks.

Acknowledgements

Over the last months I've read several "Acknowledgements" of many colleagues that got to the end of this crazy, frustrating, exciting, and surprisingly funny journey. However, I feel like this is the only space in my thesis where my love for writing can make a difference. So I excuse myself, but I hope this will be different in a good way.

To my advisor, Cyrill Stachniss, I owe the biggest "thank you". He made me grow professionally and as a person. He taught me that sometimes we lose, but that doesn't make any less of us. He showed me how to be confident, but humble. I couldn't imagine anyone better than him to guide me through such a complex journey as the one of my PhD.

To my direct supervisor, Jens Behley, I think I also owe some apologies. He always pushed me to be at my best, in every line of text, image, experiment, and idea. It was not always easy to accept his standards, especially when I just really wanted to get things done. Thanks to him I know what it means to breathe and do the things at my best, even when I don't really want to.

Thanks to Birgit Klein and Thomas Läbe, for all their support while navigating the madness of German bureaucracy, servers set up, cables, and post letters. They helped my first steps in this country and this lab and were always available for any dumb questions I had. And to be fair, I had many.

I also need to thank the PhenoRob Cluster Office for assisting me these past four years, especially Franziska Külbel. She is one of the most caring, direct, warm, and driven people I have come to know here in Bonn.

A special thank goes to all my colleagues, the ones at UC Davis who welcomed me as a long-time friend during my visiting, and the ones in Bonn, who managed to make the rule of "no remote work" enjoyable. In particular, I would have left this journey crying without some of the best friends I could ever imagine finding. Thanks, from the bottom of my heart to Linn Chong, Federico Magistri, Rodrigo Marcuzzi, Elias Marks, Lucas Nunes, Julius Rückin, Matteo Sodano, Lina Stausberg, and Louis Wiesmann. You picked up the pieces of my broken research and made me see the light on days when Bonn's sky was way too dark.

I want to thank someone who made the process of writing my thesis a lot

easier, maybe because shared pain is always less of a burden. Sara Bonini, thanks for the laughs, the shoulder pats, the hugs, and the beers, and thank you for your support. My life would have a lot more blank pages without you.

I cannot speak of friends without talking about the ones I left at home. I discovered in the last months how hard it is to be the one staying when important people leave to chase bigger things. Thanks, for never making me feel out of place, before and after this PhD journey who brought me away from you. Federica Mariconda, Mattia Seu, Valerio Nicolanti, Elisa Ciaffi, and Simone Pelonzi, you are the best thing ever happened to me. I only hope I can give you the same support and love you always showed me.

As a last thing, I have to thank my family. To my parents, I will never forget your face when you left me here on the first day. I felt like a preschool kid, and sometimes I still feel the same. You always made me think I could achieve anything, thanks for believing in me even when I was not sure it was a good idea.

To my sister Roberta, thanks for leading through example. I always looked up to you, and you never failed in teaching me how to be more compassionate, balanced, understanding, and driven. Thanks for teaching me without making me feel dumb, and for loving me even when I was really dumb. I would not be here today without you in my life.

The work presented in this thesis is partially supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy, EXC2070-390732324-PhenoRob. The financial support of the DFG through the PhenoRob project is gratefully acknowledged.

Contents

Chapter 1

Introduction

The world's population is rapidly increasing and will reach 9 billion people by 2050 according to recent United Nations projections [211]. Population growth and shrinking arable land due to urbanization and unsustainable farming increase the demand for sustainable agricultural technologies [68]. Climate change places additional pressure on sustainable food production due to altered precipitation patterns, an increase in extreme weather events, and the reduced reliability of traditional growing seasons. At the same time, the agricultural sector is facing stricter regulations concerning water usage, greenhouse gas emissions, and the impact on biodiversity loss. All these challenges together with desertification, salinization, and soil erosion, pose a limit to the common practice of expanding farmland. Thus, the problem of increasing crop production has been recently addressed by developing new crop varieties with higher yield and resistance to stress and diseases [200]. However, this alone cannot meet the increasing demands of food, feed, fuel, and fiber, highlighting the need for improved agricultural practices.

Agricultural robotic systems have the potential to tackle this issue by offering solutions to enhance the efficiency and sustainability of agricultural practices. For instance, weeding robots and aerial drones are already being deployed in the fields to reduce the pressure on seasonal workers. These systems allow farmers to reduce their reliance on chemicals and broad-spectrum herbicides [147] by enabling targeted applications based on real-time data about the crop and environmental conditions. Additionally, they can assist farmers in making informed decisions regarding irrigation and fertilization schedules [95], optimizing resource usage and minimizing the environmental impact. Moreover, autonomous platforms can operate continuously, thereby enabling scalable farming operations [14] to meet the growing global demand for food. Examples of such robotic platforms are illustrated in Fig. 1.1, showing both unmanned aerial vehicles (UAVs) and unmanned ground vehicles (UGVs) as they are deployed in agricultural fields.

Figure 1.1: Examples of robotic platforms used in agriculture. The first row shows UAVs equipped with different cameras flying over the crop fields. The second row shows UGVs deployed in the fields. In the image on the left, we can see how the area below the robot is isolated to control the light condition of the captured images.

A key factor to improve the efficiency and robustness of robotic systems in agricultural environments is their ability to understand their surroundings by interpreting sensor data, such as cameras providing color information (RGB) or light detection and ranging (LiDAR) sensors. LiDAR sensors provide accurate 3D geometry data, making them ideal for fruit picking and trait estimation. However, they typically lack the color information required for monitoring the field. On the other hand, RGB cameras can capture color, texture, and shape information, making them the most common choice for monitoring purposes. The lack of 3D data prevents their use for more articulate tasks such as harvesting, which requires accurate information about the position of the fruit to be harvested. To efficiently interpret RGB or LiDAR data, recent robotics platforms use neural networks trained to perform the desired agricultural task, e.g., segmentation of crops and weeds, instance segmentation of plants and leaves, or trait estimation. These tasks are consecutive steps of the phenotyping process, which aims at identifying the observable characteristics of plants. Semantic segmentation classifies each

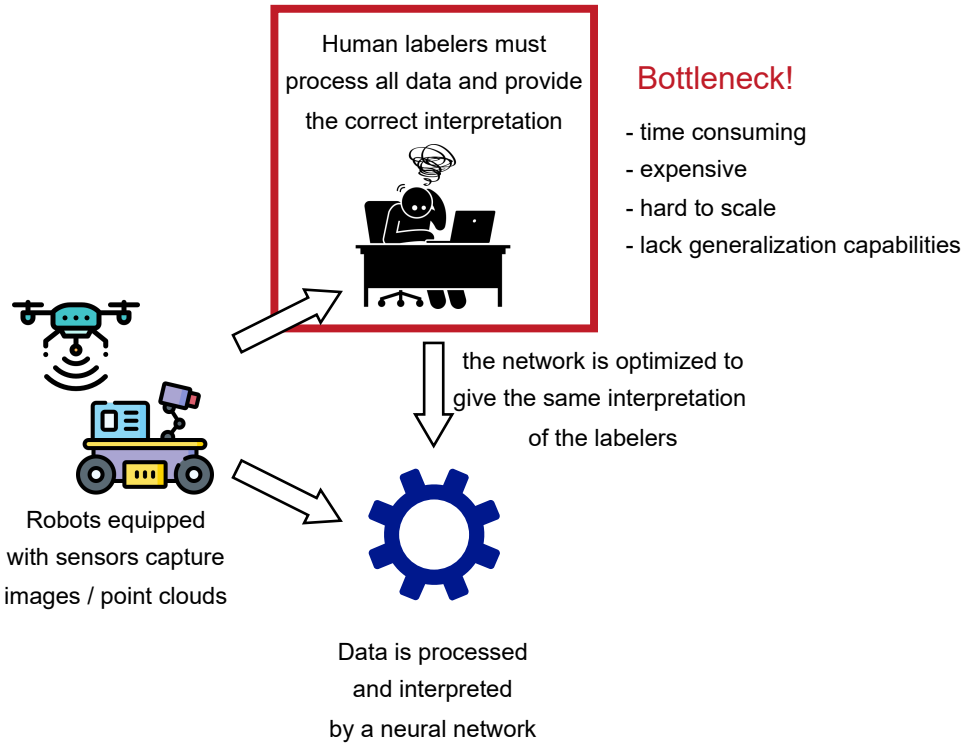


Figure 1.2: In traditional pipelines, we gather data using sensors from robotic systems, such as UAVs and UGVs. We must label all the data to train the neural network, which is optimized to produce the desired output. Labeling the data is the bottleneck of modern pipelines: it is time-consuming, expensive, and difficult to scale, and networks often can't generalize to unseen data after being trained on such annotations.

pixel in the image as soil, crop, or weed, and it is essential to enable automatic weeding. Going one step further, instance segmentation distinguishes individual instances of the same class, such as fruits, leaves, or plants. This enables fruit counting, yield estimation, growth assessment, and field monitoring. Lastly, the estimation of characteristics, such as leaf size, plant height, or fruit ripeness, is essential for precision agriculture and breeding programs.

Data-driven approaches can show convincing performance but generally need vast amounts of labeled data to achieve satisfactory performance. Data labeling is a central part of the workflow of data-driven methods: it requires human workers to analyze the sensor data and manually produce the supervision to train the deep-learning approaches. The high cost and time required for labeling data are bottlenecks in machine-learning approaches. We illustrate in Fig. 1.2 the general framework of fully supervised methods that require collecting the data, having human labelers providing the correct interpretation for all the data, and

then training the network on such annotated datasets. Annotating 3D point clouds usually takes hours for one single scan, since the difficulty of labeling in three dimensions using 2D visualization tools adds up to the difficulty of the task. However, labeling high-resolution RGB images is not less of a burden. Bosilj et al. [20] report an average of three hours to label one single image for semantic segmentation, while Weyler et al. [224] an average of two hours and a half per image. Chebrolu et al. [32] report that annotators spent over 100 hours labeling, and yet they still encountered generalization issues when they deployed the model trained on their annotated dataset. It is clear that the costs scale poorly considering that when field conditions, crop growth stages, and sensor setups change the models still need to be re-trained.

In the literature, several paradigms have been proposed to reduce the cost and labor associated with annotating datasets. One prominent approach is weakly-supervised learning [245], which relies on sparser or incomplete annotations, such as image-level labels [116], bounding boxes [91], or scribbles [242] instead of expensive pixel-wise annotations. This technique reduced the annotation effort while being compatible with common supervised learning techniques. A more radical strategy is self-supervised learning [103], which aims to eliminate the annotations from the paradigm designing tasks where the supervision can be derived from the data itself. Once trained in such a way, the models usually need to be fine-tuned over smaller labeled datasets to learn how to perform the final task. Another widely studied paradigm is active learning [16], which has been used to distinguish the most informative samples to annotate from a bigger pool of unlabeled data. This reduces the number of total labeled examples required to achieve the same performance on the task. More recently, the rise of generative approaches, such as diffusion models [46] and Generative Adversarial Networks [114], has enabled the creation of synthetic datasets. Such approaches generate high-quality data along with their corresponding annotations, reducing or even eliminating the need for manual annotations.

However, most of the existing approaches are designed for general-purpose computer vision tasks in everyday scenes or autonomous driving datasets. Applying them directly to the agricultural domain often performs poorly due to the large domain gap and the unique challenges posed by the agricultural environment [104], such as crop variability, occlusions, and field-specific patterns. Nevertheless, exploiting the contextual knowledge about agricultural fields – crop type, growth stage, field layout, use of herbicides – offers the potential to improve automatic labeling [131] or self-supervised learning techniques [187].

In this thesis, we tackle the problem of reducing the requirement for labeled data while improving the performance of the tasks related to crop monitoring in agricultural fields. Crop monitoring involves four key tasks: (i) distinguishing

crops and weeds; (ii) segmenting crops into individual plants; (iii) segmenting plants into individual leaves; and (iv) estimating leaf traits such as width and length. These tasks are increasingly challenging to perform for a robot. Crops and weeds can be very similar, and it is often hard to provide enough labeled weeds for data-driven approaches since they are usually smaller and less present in managed fields. The definition of a weed also varies by field; a plant considered a weed in one setting may be a crop in another. Additionally, identifying crops may be challenging since plants exhibit significant variations in size, shape, and structure depending on the growth stage and environmental conditions. Data-driven approaches often require retraining on labeled images of the new field, growth stage, or environmental condition before deployment. In dense crop fields of late growth stages, single plants are often difficult to identify because of occlusions caused by the same or a neighboring plant. Moreover, crops may have complex canopies that obscure individual plant boundaries. This affects both the plant and leaf instance segmentation, leading to errors in sequent downstream tasks like trait estimation and phenotyping.

Throughout this thesis, we propose new techniques to reduce the need for labeled data for agricultural robotic systems with the final objective of improving the capabilities of their perception systems. This includes domain-specific techniques to improve the instance segmentation performance on plant and leaf instances, techniques to provide better initialization for networks performing semantic and instance segmentation in 2D and 3D scenarios, techniques to improve the automatic labeling of agricultural images for semantic and plant instance segmentation, and finally techniques to generate synthetic labeled data to improve the performance of trait estimation methods.

1.1 Main Contribution

The main contribution of this thesis is a set of novel techniques to improve the semantic scene understanding in agricultural fields with the goal of reducing the requirements for manually annotated data. Our proposed techniques range from domain-specific pre-training for network initialization, to unsupervised label generation to train supervised approaches for agricultural perception tasks.

In Chapter 3 we present an approach to jointly perform semantic, plant instance, and leaf instance segmentation of RGB images, exploiting the hierarchical structure of the tasks and automatic domain-specific post-processing to enhance our performance. In our experiments, we demonstrate the effectiveness of our approach on multiple datasets of different crop species. Within this thesis, we mainly focus on the domain-specific post-processing, that allows us to outperform state-of-the-art approaches and improve our predictions even if applied in

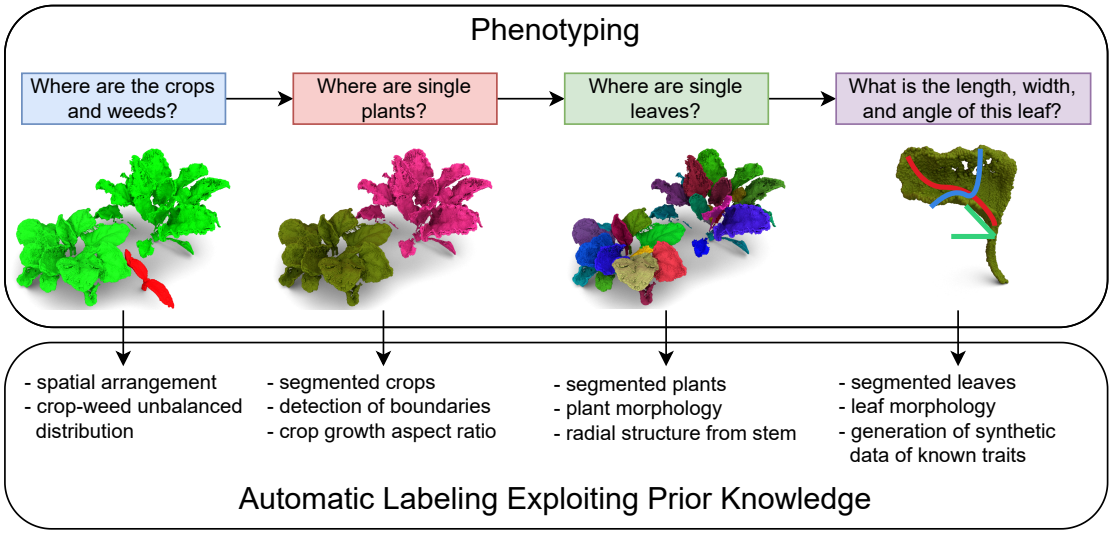


Figure 1.3: This thesis addresses the problem of reducing or automatizing the labeling required for different phenotyping tasks. More specifically, in Chapter 3, Chapter 4, and Chapter 7 we improve the performance of semantic, plant instance and leaf instance segmentation reducing traditional labeling; in Chapter 5 and Chapter 6 we focus on automatic labeling for semantic and plant instance segmentation; finally, in Chapter 8 we show how to automatically generate labeled data for leaf trait estimation.

the context of fully supervised training. This raises the question of how to reduce the requirement for annotated data by exploiting more of the domain-specific knowledge we have about agricultural fields.

As a first way to answer this question, in Chapter 4 we present a set of novel augmentation techniques for agricultural images that enables a domain-specific pre-training. Our pre-training allows us to exploit the large amounts of unlabeled data captured on agricultural fields to effectively pre-train deep-learning approaches for different perception tasks reducing the annotated data required to reach the same performance to 25% of the original dataset size. We investigate the importance of each augmentation and their application's order, comparing against commonly used general-purpose large pre-training datasets. Our techniques boost the results over all the investigated scenarios and produce a valuable initialization technique for deep-learning approaches in the agricultural domain. In order to have a common pre-training for different perception tasks, we are not able to use all of our prior knowledge, e.g. information about the spatial arrangement of the field is crucial for semantic segmentation but not for leaf instance segmentation. Thus, we decide to focus on single tasks to better exploit all relevant information of the fields.

We proceed following the phenotyping tasks illustrated in Fig. 1.3, starting from answering the question "Where are the crops and weeds?", i.e. performing

semantic segmentation. In Chapter 5 we present our approach to automatically generate semantic labels for posed RGB images. We use the posed images to enhance the spatial consistency of the generated labels, enabling the detection of multiple crop rows in single images. We leave the areas where we are most likely to commit errors unlabeled. This, coupled with an uncertainty-aware deep-learning framework and the knowledge of the class imbalance between crops and weeds, enables us to boost the performance of our semantic segmentation without the requirement for manually annotated images. We experimentally demonstrate that our approach can produce reliable semantic labels on different crop species and that using our annotated datasets can improve the generalization capabilities of fully supervised learning approaches.

We then tackle the problem of separating the crops into single instances, i.e., performing plant instance segmentation. The fourth contribution of this thesis, presented in Chapter 6, is a domain-specific instance segmentation technique to obtain plant instances from a partial instance segmentation that can be provided by large general-purpose vision-language models or from graph-based heuristic methods. Our approach uses domain knowledge to improve the performance of the partial instance segmentation, allowing the heuristics-based method to outperform all other heuristics-based approaches, and improving the predictions of the vision-language models without the need for further annotated images. Our experiments show that our domain-specific method produces accurate plant instances for different crop species without the need for extensive training or manual labels. We also show different strategies to use our generated labels to boost and improve the generalization capabilities of fully supervised learning methods while reducing the requirement for labeled data.

While all the contributions up to this point allow performing segmentation tasks crucial for plant phenotyping, most of the robotics platforms need access to 3D information to be able not only to analyze but also to intervene in the fields. On top of that, the lack of depth information in 2D images prevents the extraction of accurate phenotypic traits. Thus, we decided to perform leaf instance segmentation in the 3D domain. In Chapter 7 we present our pre-training approach for 3D leaf instance segmentation. We pre-train our network for the leaf instance segmentation task, optimizing the network for a similar task, exploiting our knowledge about the plant structure. Additionally, we propose novel 3D data augmentation to enhance the robustness of our approach against occlusions caused by other leaves and plants, and distortions caused by the wind. Our experiments confirm that our pre-training improves the results while reducing the need for labeled data.

The 3D leaf instance segmentation is the last crucial step that allows for the estimation of leaf traits, such as the leaf blade length and width. These

traits are linked to crop growth, productivity, and pest resistance. Our last contribution presented in Chapter 8, is an approach to automatically generate a labeled dataset of leaf point clouds and associated leaf blade length and width. We use this dataset to fine-tune several off-the-shelf trait estimation approaches. Our experiments validate that our generated dataset allows for more accurate trait estimation of real-world data compared to datasets generated by other means or manual measurements of the traits that are usually per plot, i.e., per sub-area of the field and not for each leaf.

To summarize, we propose several novel methods to improve the vision capabilities of robotic systems performing phenotyping tasks in agricultural environments. In Fig. 1.3, we present a schematic overview of the contributions of this thesis, focusing on how we exploit prior domain knowledge to reduce the need for manually annotated datasets or to tackle the problem of generating such labels in an automatic fashion.

1.2 Publications

Parts of this thesis have been published in the following peer-reviewed conference and journal articles:

- G. Roggiolani, F. Magistri, T. Guadagnino, J. Weyler, G. Grisetti, C. Stachniss, and J. Behley. On Domain-Specific Pre-Training for Effective Semantic Perception in Agricultural Robotics. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2023. DOI: 10.1109/ICRA48891.2023.10160624.
- G. Roggiolani, M. Sodano, F. Magistri, T. Guadagnino, J. Behley, and C. Stachniss. Hierarchical Approach for Joint Semantic, Plant Instance, and Leaf Instance Segmentation in the Agricultural Domain. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2023. DOI: 10.1109/ICRA48891.2023.10160918
- G. Roggiolani, F. Magistri, T. Guadagnino, J. Behley, and C. Stachniss. Unsupervised Pre-Training for 3D Leaf Instance Segmentation. *IEEE Robotics and Automation Letters (RA-L)*, 8(11):7448–7455, 2023. DOI: 10.1109/LRA.2023.3320018
- G. Roggiolani, J. Rückin, M. Popović, J. Behley, and C. Stachniss. Unsupervised Semantic Label Generation in Agricultural Fields. *Frontiers in Robotics and AI*, 12:1548143, 2025. DOI: 10.3389/frobt.2025.1548143
- G. Roggiolani, B. N. Beiley, J. Behley, and C. Stachniss. Generation of Labeled Leaf Point Clouds for Plants Trait Estimation. *Plant Phenomics*, 7(3):100071, 2025. DOI: 10.1016/j.plaphe.2025.100071.
- G. Roggiolani, J. Behley, and C. Stachniss. Plant-Specific VLMs Refinement for Plant Instance Segmentation. *under review at Information Processing in Agriculture*

Additionally, I have been involved in the following data publications during my doctorate together with other researchers. This dataset paper is not directly part of this thesis, but has been used in the experimental evaluations:

- J. Weyler, F. Magistri, E. Marks, Y.L. Chong, M. Sodano, G. Roggiolani, N. Chebrolu, C. Stachniss, and J. Behley. Phenobench: A large dataset and benchmarks for semantic image interpretation in the agricultural domain. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, 46(12):9583–9594, 2024

1.3 Open-Source Contribution

With the idea of facilitating reproducible and unbiased evaluation of new research ideas, in addition to the previously mentioned journal and conference publications, we made several open-source contributions including datasets and source code of our implementations:

- The code for our hierarchical joint semantic, plant, and leaf segmentation is available at:
  `https://github.com/PRBonn/HAPT`,
- The code for our domain-specific pre-training is available online at:
  `https://github.com/PRBonn/agri-pretraining`,
- The code for our unsupervised semantic segmentation pipeline is available online at:
  `https://github.com/PRBonn/umsemlabag`,
- The code for our unsupervised plant instance segmentation is available online at:
  `https://github.com/PRBonn/PlantInstGen`,
- The code for our 3D leaf instance segmentation pre-training is available online at:
  `https://github.com/PRBonn/Unsupervised-Pre-Training-for-3D-Leaf-Instance-Segmentation`,
- The code for generating labeled leaf point clouds for traits estimation is available online at:
  `https://github.com/PRBonn/3DLeafLabGen`
- The semantic, plant, and leaf instance segmentation dataset PhenoBench is available online at:
  `https://www.phenobench.org/`.

Chapter 2

Related Work

Robotic applications in agriculture aim at improving field monitoring and interventions diminishing the use of agricultural chemical inputs and production costs [172, 214]. Over the last years, we have seen significant progress in the application of vision-based methods for several tasks in real agricultural settings, such as semantic segmentation, instance segmentation, and trait estimation. The task of semantic segmentation requires providing a pixel-wise classification of the input image [47]. In the agricultural setting, this could mean assigning a class between soil, crop, or weed to each pixel. The next step is to separate the pixels of each class into individual instances, assigning a unique identifier to each object in the scene [75], for example to every single plant or leaf. Both tasks are critical for helping farmers and breeders gain a better understanding of the crop phenotype in their fields. Phenotyping refers to the computation of traits of the crops, such as the plant height, the length and width of leaves, the number of leaves, or the plant density [113, 220]. Only a deep semantic understanding of the agricultural field enables the estimation of relevant traits used by farmers and breeders for in-field intervention [112], potentially by a robotic system [171].

The first visual methods for segmentation were heuristics-based approaches for images, leveraging statistical and geometrical properties to separate the image into semantically related regions. One of the earliest methods was proposed by Otsu et al. [164], where they segment the image into foreground and background using an automatic threshold. Several approaches [23, 158, 170] start from user-defined initial seeds and expand them in regions of similar features, allowing for segmenting multiple areas in the same image. To overcome the need for initial seeds, Lloyd et al. [135] as well as Tomita et al. [206] use statistical analysis over the image to capture the similarity of groups of pixels. One line of works focuses on adaptive thresholding [53, 111, 235] based on the lighting condition and the histogram distribution of the image, and then on the use of multiple thresh-

olds [173, 177, 218] to perform object-level segmentation. A second set of works instead relies on detecting edges [25, 29, 33] and treats them directly as object boundaries. More complex heuristics-based approaches exploiting geometric cues construct graphs over the image [62] and take into consideration the global properties of the image to guide region merging and splitting. Heuristic approaches are also applied for the segmentation of 3D data [21, 168], exploiting the knowledge about the plants' structure to separate a single plant from the ground and neighboring overlapping plants. Miao et al. [148] propose to extract the plant's skeleton, run a first coarse segmentation, and then perform a fine segmentation based on morphological features. Jin et al. [102] follow a similar approach, using a growing algorithm to segment the stem after removing the ground points. However, purely heuristics-based pipelines often depend on several hand-tuned parameters and domain knowledge, leading to poor generalization performance. In the specific case of the agricultural domain, heuristics-based pipelines struggle with the problem of occlusions caused by neighboring plants or even the plant itself, failing to identify correct instances in case of high plant density.

With higher computational power and the collection of larger datasets [106, 224], deep-learning approaches have become the dominant way to solve segmentation tasks. They usually outperform traditional heuristics-based methods, especially in complex scenarios where manually tuned heuristics are demanding to design. Recent advances primarily leverage convolutional neural networks (CNNs), vision transformers (ViTs), and more recently vision-language models (VLMs) to extract rich feature representations for segmentation [26, 83, 184].

CNN-based architectures such as U-Net [186] and DeepLabV3 [34] are commonly employed for pixel-wise classification in agriculture due to their ability to capture multi-scale features and retain spatial information. Cui et al. [48] introduced an improved U-Net variant for corn and weed segmentation, while Zenkl et al. [240] utilized DeepLabV3 to segment wheat leveraging features at different image resolutions via atrous spatial pyramid pooling. Zeng et al. [239] proposed a framework to estimate the true near-infrared (NIR) reflectance of vegetation compensating for shadows, point of view, and soil conditions. A correct NIR reflectance is crucial for approaches such as the one by Milioto et al. [150], where they compute multiple vegetation indices using NIR and RGB images to enhance the segmentation accuracy of neural networks. Beyond classical CNNs, Weyler et al. [225] employed ERFNet [184] for efficient real-time segmentation embedded in robotic systems. ERFNet combines residual skip connections [84] and factorized convolutions [217] to maintain high accuracy while reducing the computation load, making it well-suited for in-field robotic systems.

Once each pixel in the image is classified, the next step is to distinguish individual objects within the same class. For the instance segmentation task in

the agricultural domain, Champ et al. [30] investigated one of the most common instance segmentation approaches, Mask R-CNN [83]. It follows a two-stage pipeline: object detection followed by pixel-wise mask generation. Another notable method, PanopticDeepLab [39], formulates instance segmentation by predicting object centers and pixel-wise offsets, followed by post-processing to group pixels into individual instances. Weyler et al. [222] adopted a similar approach but introduced covariance matrices to refine instance clustering, particularly for leaf segmentation. Morris [155] designed a pyramid CNN that distinguishes leaf boundaries based on texture differences, and Romera-Parades et al. [185] introduced convolutional long short-term memory (LSTM) units to count leaves exploiting their spatial layout. Beyond single-network architectures, ensemble and multi-branch models have also been explored. Jeon et al. [98] proposed a dual-network approach, where different feature representations are exchanged during training, resulting in an ensemble-based final prediction.

Recent advances in vision models have significantly impacted the performance of semantic and instance segmentation in a large variety of domains, agriculture included. The Segment Anything Model (SAM) [110] was trained on over 1 billion masks of general-purpose data and showed impressive generalization capabilities in several domains. The possibility to prompt it with bounding boxes, points, or masks enables the segmentation of crops, plants, or even plant organs. Then, vision-language models such as Grounded SAM [178] included the possibility of prompting the segmentation pipeline with natural language. While the performance of these models is impressive, their reliability falls short in the agricultural domain due to the presence of occlusions and the high variance in the appearance of soil, crops, and weeds. Fine-tuning them on manually labeled datasets is still the go-to solution for achieving satisfactory performance.

Despite the amount of work and progress in 2D segmentation, many agricultural applications require a more detailed understanding of the plant structure which can be achieved only using 3D data. In particular, the estimation of morphological traits often requires spatial reasoning beyond the capabilities of images. As a result, there is an increasing trend of works based on three-dimensional data, which can capture the complex geometry of plant structures. However, 3D representations pose challenges due to their irregular and unordered nature. In the work by Ao et al. [9], they use PointCNN [125] to extract stem points, and then fit them into 3D cylinders. Individual leaves are segmented using a density-based clustering algorithm [59], followed by morphological post-processing. Similarly, Han et al. [78] proposed to predict object centers and cluster the instances using the mean-shift algorithm [45]. Li et al. [123] proposed a novel down-sampling strategy to preserve edge points and a novel fusion mechanism for semantic and instance features, enabling the networks to identify instance boundaries and im-

prove their performance. They further improved their method in their following work [122], adding modules to exploit the attention mechanism [213] and for fusing features at different resolutions.

While deep-learning approaches do not suffer from the limitations of the heuristic-based methods, they require vast amounts of manually annotated data to achieve a satisfactory performance. Data labeling is the bottleneck of such methods since it is expensive and requires expert knowledge to obtain highly accurate datasets. One common way to reduce the need for annotated data is to pre-train the network, reducing also the time needed to converge. In the literature, pre-training on large labeled datasets [31, 176] is a common choice for initializing the network for various tasks and domains. However, pre-training on large general-purpose datasets does not provide a good initialization when the application domain is narrow and distant from the original training dataset. Self-supervised pre-training methods, which train models without labeled data by leveraging task- or domain-specific constraints, have shown promise in reducing annotation needs while performing on par with fully supervised pre-trainings on different tasks [37, 38, 82, 238]. One of the main lines of work for self-supervised pre-training focuses on contrastive learning approaches, which augment the input to produce different views aiming to obtain similar features for views of the same sample. Researchers have also explored various data augmentation techniques to produce different views, showing the relevance of each augmentation and the order dependence of their applications.

Xie et al. [189] adapted the contrastive learning paradigm used for images [37, 238] to the point cloud domain, encouraging representations of augmented views of the same object to be similar, while pushing apart representations of different objects. Building on this, Zhang et al. [243] introduced a momentum encoder to maintain a more consistent batch of negative examples. Other methods explored the possibility of pre-text tasks that don't require labels and encouraged the network to learn spatial and geometric features from the data. Wang et al. [215] simulate occlusions and require the network to reconstruct the missing section, while Alliegro et al. [8] propose a 3D jigsaw puzzle where point cloud segments are shuffled and the network must learn to assemble them back in the original shape. Achituve et al. [2] went in the direction of point-wise manipulations, displacing points of the input and letting the network predict their original locations. Pre-training provides strong initial representations, but the networks still need fine-tuning on data labeled for the target task.

Another promising strategy to reduce the reliance on manual labels is to automatically generate labels integrating domain-specific priors. In the agricultural domain, Lottes et al. [133] exploit the field arrangement to detect crop rows for effective semantic segmentation, and Winterhalter et al. [226] further improve

their performance by looking for multiple parallel crop rows at once. Despite the efforts, crop-row-based segmentation approaches rely on simplifying assumptions that do not always hold in real-world scenarios. In practice, weeds frequently grow within the crop rows, and overlapping plants introduce visual ambiguity, particularly in late growth stages. To maximize the accuracy of the generated labels, a line of works focuses on synthetic data generation [27, 124]. Exploiting the knowledge of expert plant scientists, these works build 3D scenes of the agricultural fields [11, 86]. Generated data directly comes with labels for semantic and instance segmentation, as well as other complex annotations, such as phenotypic traits, that can be exploited to fine-tune and train any geometric or deep-learning approach. Nevertheless, these methods require significant expert knowledge. Such generation often relies on a mechanistic plant model or handcrafted rules. As a result, they need to be adapted for any new crop species or different growth stages, limiting their scalability. Data-driven generative approaches such as Generative Adversarial Networks [70] and Diffusion models [46] try to address this limitation, synthesizing novel and realistic training data without the need for manual modeling. Generative Adversarial Networks produce high-fidelity images but they often suffer in capturing diverse or fine-grained details [219], which can be crucial in differentiating plant species, growth stages, and single plant organs. Diffusion models use a denoising process to generate diverse samples while preserving semantic consistency, but they still struggle to generate outputs without careful conditioning [6].

In this thesis, we propose different methods to enhance the performance of neural networks on the perception tasks in agricultural fields exploiting prior knowledge specific to the agricultural domain. We build upon previous advances in self-supervised learning by designing agricultural-specific augmentation strategies and pre-training tasks tailored to plant phenotyping. These strategies improve results compared to commonly used pre-training strategies that do not take into consideration the application domain. By leveraging intrinsic field priors, such as crop row arrangement and typical plant density and morphology distributions, we enable fully unsupervised semantic and instance segmentation for automatic labeling of images. Our methods generate high-quality labeled data without relying on costly manual annotations or assumptions that are easily violated in the fields. Finally, we propose a method to generate labeled leaf point clouds for the estimation of morphological traits, such as leaf blade length or width. Our method relies on real-world data and does not require manual annotation or expert knowledge. Our method generates a dataset with accurate measurements and sufficient variety to fine-tune off-the-shelf trait estimation methods improving their final performance on real-world measured traits.

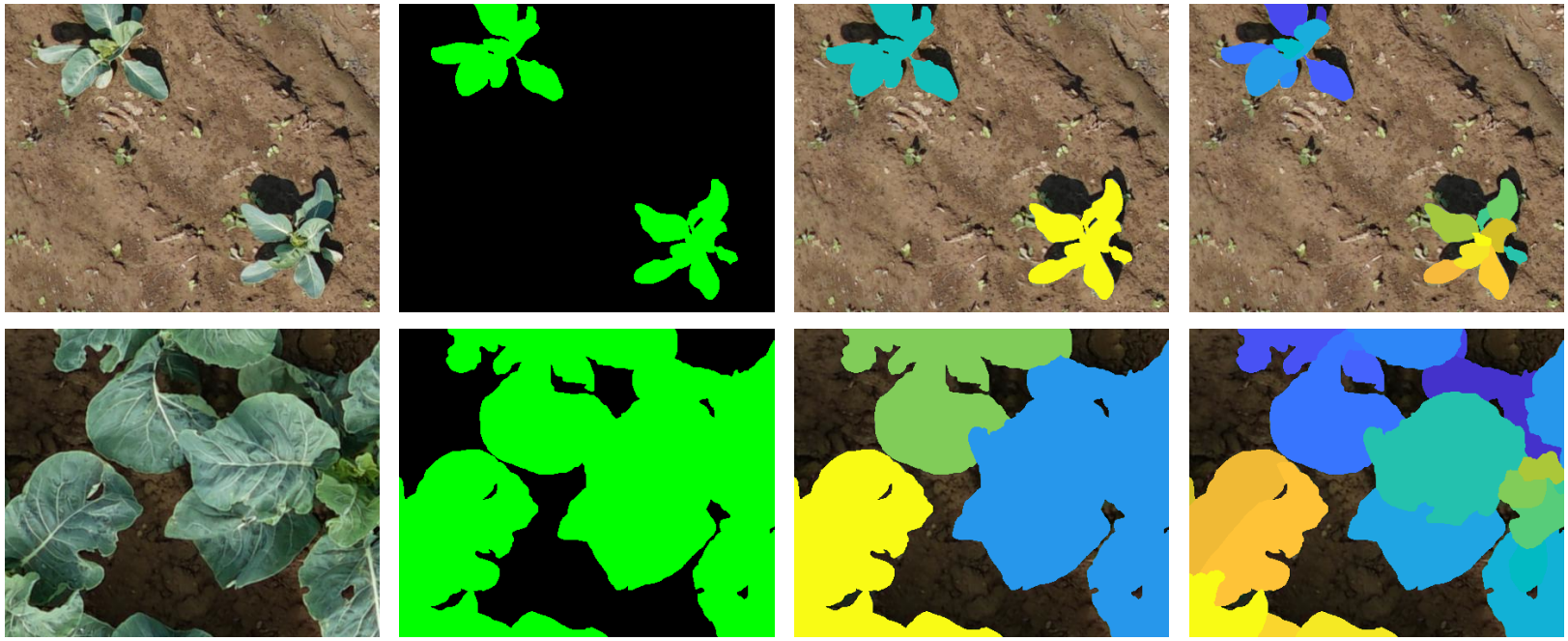
Chapter 3

Exploiting Domain Knowledge for Post-Processing of Instances

Sustainable crop farming is crucial to fulfill the demand for food, fuel, and fiber while reducing the environmental impact of agricultural practices. To this end, plant phenotyping aims to accurately identify plant growth stages and appearance to optimize field management or provide variety-specific information to plant breeders [204]. The first step towards phenotyping is the accurate identification of crops and weeds, which can be automated by robotic platforms equipped with sensors. Additionally, robotic platforms can leverage information about the plant's growth and phenotypic traits to perform automatic in-field intervention. One commonly analyzed phenotypic trait is the number of leaves per plant, which is a key component for assessing the growth stage and the need for fertilization [118].

In this chapter, we propose a solution to simultaneously perform semantic, plant instance, and leaf instance segmentation of crops given an RGB image recorded by UAVs. In Fig. 3.1, we show two exemplary images of cauliflowers from the GrowliFlower [106] public dataset with their corresponding semantics, where crops are depicted in green, and instances of plants and leaves, where each color represent a different and unique instance. The three segmentation tasks are interdependent, but prior vision-based approaches targeted semantic segmentation of crops and weeds [145, 150], instance segmentation of plants [30] or instance segmentation of leaves [222, 225] in isolation. Consequently, these approaches overlook the underlying hierarchical relationship between these tasks, where semantic knowledge informs plant identification, which guides the detection of leaf boundaries. Our method explicitly leverages this task hierarchy enabling more accurate and coherent results.

The contributions of this chapter are twofold: (i) a novel CNN-based network architecture designed to jointly perform semantic segmentation and instance seg-



RGB Semantics Plant Instances Leaf Instances

Figure 3.1: Our approach takes as input RGB images (first column) from real fields and provides semantic (second column) where green is the vegetation and black is the soil, plant instance (third column), and leaf instance segmentation (last column). Different instances are shown with different colors.

mentation of both plants and leaves; and (ii) a domain-specific post-processing that significantly boosts segmentation performance by refining the raw network output using agricultural priors. While this work is part of a shared publication, the design and implementation of the post-processing pipeline is my principal contribution, and experimental validation was a joint effort. We present both contributions here for a complete understanding of the system and its results. For each pixel in the input image, we predict its semantic class and, if it is a crop to which plant and leaf instance it belongs. We solve the three tasks jointly, exploiting the underlying task hierarchy thanks to a novel design of residual skip connections. In particular, semantic segmentation can support plant instance segmentation, which can further guide leaf instance segmentation. Furthermore, our post-processing module exploits structural regularities in crop layout and morphology to enable high-quality instance segmentation. Our approach yields a pixel-wise semantic, plant instance, and leaf instance segmentation of the image data at the frame rate of a typical camera, i.e., approx. 25 frames per second. Our experiments suggest that our novel skip connections scheme better exploits the hierarchical connections between tasks, and our automatic post-processing achieves superior performance compared to common state-of-the-art methods while yielding end-to-end real-time inference.

3.1 Our Approach for Hierarchical Image Segmentation in Agriculture

The network that we propose is an encoder-decoder architecture with three decoders, addressing the three tasks of semantic, plant instance, and leaf instance segmentation, as illustrated in Fig. 3.2. The network takes as input RGB images $\mathbf{I} \in \mathbb{R}^{3\times H\times W}$, where H and W are, respectively, the image height and width, and 3 is the number of channels of a typical RGB image. We employ an ERFNet [184] encoder and three ERFNet-based decoders. This choice provides us with a lightweight network well-suited for tasks requiring real-time predictions. The semantic segmentation decoder consists of a single non-bottleneck-1D block after the deconvolutions, while the instance segmentation decoders have two, as defined in the original paper [184]. Both encoder and decoders use the Gaussian error linear unit (GELU) [87] activation function, a smoother variant of the common Rectified Linear Unit (ReLU), following the suggestions by Liu et al. [129].

3.1.1 Decoders for Semantic and Instance Segmentation

Semantic segmentation. The decoder for semantic segmentation has a single output head, and the output $\hat{\mathbf{I}}_{\text{sem}}$ has size $H \times W \times K$, where K is the number

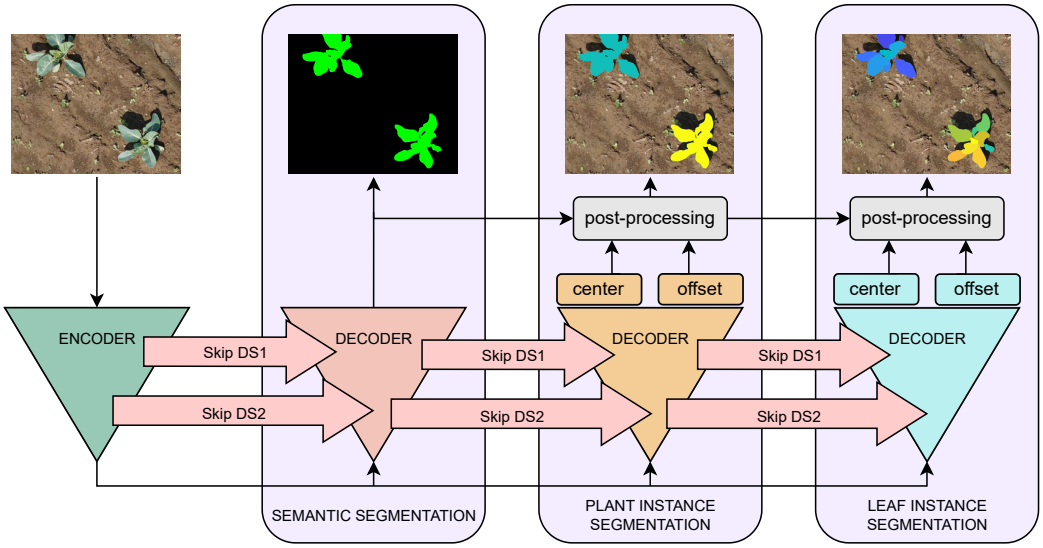


Figure 3.2: Overview of our architecture. The encoder takes an RGB image as input and extract compressed features that are further elaborated by the decoders. Residual kip connections are present in a hierarchical fashion after each downsampling (encoder) and upsampling (decoders) block. The two features map that are passed via the skip connections have size $H/2 \times W/2 \times 64$ and $H/4 \times W/4 \times 128$, respectively. The decoders for instance segmentation output predictions for centers and offsets that must be post-processed to obtain the final instance masks.

of semantic classes. In our experiments $K = 2$, since the datasets provide labels for soil and vegetation. We apply a softmax activation function to the output to obtain per-class probabilities for each pixel. We train the semantic segmentation decoder to minimize the Lovasz-Softmax loss [17], defined as

$$\mathcal{L}_{\text{sem}}\left(\hat{\mathbf{I}}_{\text{sem}}, \mathbf{I}_{\text{sem}}\right) = \frac{1}{K}\sum_{k=0}^{K-1} \text{Lovász}\left(\mathbf{L}_k\right), \tag{3.1}$$

where $\mathbf{I}_{\text{sem}}$ is the ground truth semantic, $\text{Lovász}\left(\cdot\right)$ is the application of the Lovász extension to directly optimize the Intersection-over-Union, and $\mathbf{L}_k$ is the per-class error vector defined as

$$\mathbf{L}_k = \begin{cases} 1 - \hat{\mathbf{I}}_{\text{sem},k} & \text{, where } \mathbf{I}_{\text{sem}} = k \\ \hat{\mathbf{I}}_{\text{sem},k} & \text{, otherwise} \end{cases}, \tag{3.2}$$

where $\hat{\mathbf{I}}_{\text{sem},k} \in \mathbb{R}^{H \times W}$ is the predicted probability for class k.

Instance segmentation. The two decoders for plant and leaf instance segmentation both have two heads, one to predict centers and one to predict offsets. The center prediction heads have an output of dimension $H \times W \times 1$, that is passed through a sigmoid activation function to predict pixel-wise probabilities

of being a center $\hat{\mathbf{I}}^i_{\text{cen}},\ i \in \{p,\, l\}$, where p stands for plants and l for leaves. We define the center of an object as the internal pixel closest to its median point. We optimize the center prediction to minimize the binary focal loss [151], defined as

$$\begin{aligned}\mathcal{L}^i_{\text{cen}}\left(\hat{\mathbf{I}}^i_{\text{cen}}, \mathbf{I}^i_{\text{cen}}\right) = &- \mathbf{I}^i_{\text{cen}}\left(1 - \hat{\mathbf{I}}^i_{\text{cen}}\right)^{\gamma_{\text{cen}}} \log\left(\hat{\mathbf{I}}^i_{\text{cen}}\right) \\ &- \left(1 - \mathbf{I}^i_{\text{cen}}\right)\left(\hat{\mathbf{I}}^i_{\text{cen}}\right)^{\gamma_{\text{cen}}} \log\left(1 - \hat{\mathbf{I}}^i_{\text{cen}}\right),\end{aligned} \tag{3.3}$$

where $\mathbf{I}^i_{\text{cen}}$ is the image of the ground truth centers, and γ_{cen} is a parameter to weight the loss differently in the presence of hard examples. We keep $\gamma_{\text{cen}} = 2$, which is the default value, while computing the losses for the centers of both plants and leaves.

The offset prediction heads have an output of dimension $H \times W \times 2$, since they predict offset images $\hat{\mathbf{I}}^i_{\text{off},0}$ and $\hat{\mathbf{I}}^i_{\text{off},1}$ in the x and y directions respectively. We optimize the offsets to minimize L1 losses

$$\mathcal{L}^i_{\text{off}}\left(\hat{\mathbf{I}}^i_{\text{off}}, \mathbf{I}^i_{\text{off}}\right) = \left|\hat{\mathbf{I}}^i_{\text{off},0} - \mathbf{I}^i_{\text{off},0}\right| + \left|\hat{\mathbf{I}}^i_{\text{off},1} - \mathbf{I}^i_{\text{off},1}\right|, \tag{3.4}$$

where $\mathbf{I}_{\text{off}}$ is the image of the ground truth offsets.

Thus, the final loss function $\mathcal{L}$ is given by

$$\begin{aligned}\mathcal{L} = &\, w_1\, \mathcal{L}_{\text{sem}}\left(\hat{\mathbf{I}}_{\text{sem}}, \mathbf{I}_{\text{sem}}\right) \,+\, w_2\, \mathcal{L}^p_{\text{cen}}\left(\hat{\mathbf{I}}^p_{\text{cen}}, \mathbf{I}^p_{\text{cen}}\right) \,+\, w_3\, \mathcal{L}^l_{\text{cen}}\left(\hat{\mathbf{I}}^l_{\text{cen}}, \mathbf{I}^l_{\text{cen}}\right) \\ &+\, w_4\, \mathcal{L}^p_{\text{off}}\left(\hat{\mathbf{I}}^p_{\text{off}}, \mathbf{I}^p_{\text{off}}\right) \,+\, w_5\, \mathcal{L}^l_{\text{off}}\left(\hat{\mathbf{I}}^l_{\text{off}}, \mathbf{I}^l_{\text{off}}\right),\end{aligned} \tag{3.5}$$

where w_i are scalar weights for the different terms.

3.1.2 Skip Connections

Skip connections are fundamental in several architectures [84, 92, 186] to ensure feature reusability and solve the gradient degradation problem of deep models. They skip one or more layers and provide a direct gradient flow from late to early stages. This helps maintain stronger gradient signals during training, which is a common challenge for deep networks [199]. Skip connections preserve low-level spatial information usually lost during the downsampling operations performed in the network's encoder. The common usage of skip connections in segmentation models is inspired by Ronneberger et al. [186], where the higher-resolution feature maps of the encoder are concatenated with the feature maps of the decoder. In this work, we propose a new scheme for residual skip connections, i.e., skip connections that are summed instead of concatenated, to leverage the semantic relations between the different tasks we address. We propose to directly connect the different decoders, rather than encoder and decoders only, to improve segmentation performance. Our skip connections propagate feature maps

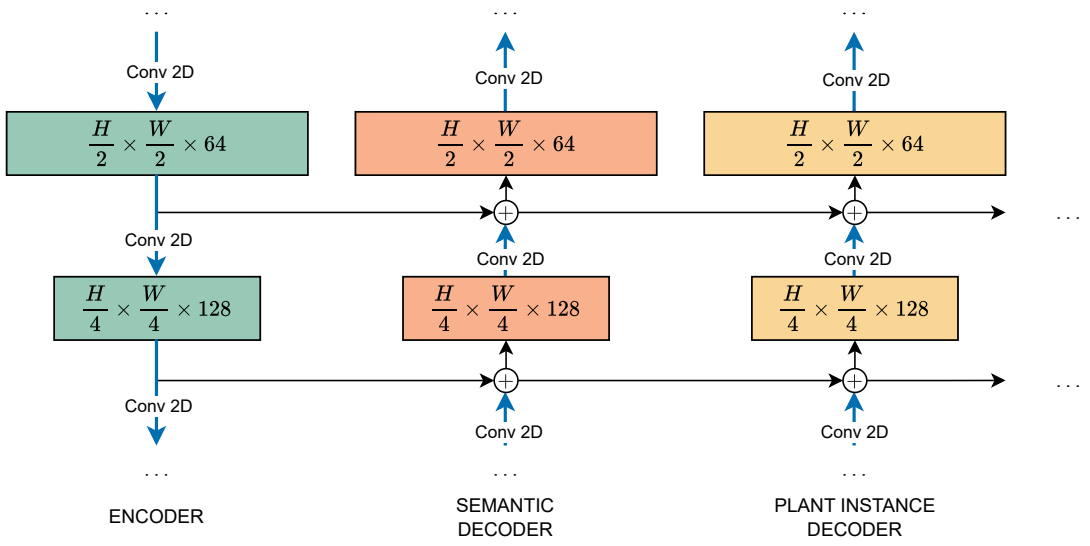


Figure 3.3: Detailed overview of our residual skip connections scheme. We take the features extracted from the convolutional layers of the encoder and sum them to those extracted from the layers of the semantic decoder. The result of this sum is passed to the following layers of the semantic decoder, and is also summed to the features extracted from the plant instance decoder. The same pattern is replicated from the plant instance decoder to the leaf instance decoder.

of dimensions $H/2 \times W/2 \times 64$ and $H/4 \times W/4 \times 128$, as shown in Fig. 3.3. In particular:

1. For the semantic segmentation task, we keep the commonly used skip connections from the encoder to the decoder [186], since spatial information coming from the features extracted at higher resolutions helps the decoder correctly classify each pixel.

2. Plant instance segmentation aims to assign each pixel classified as the crop class to a specific instance, so the information coming from the semantic task is relevant to focus on the regions of interest. The information coming from the encoder could help, but it would carry less semantic knowledge. Thus, we sum the contribution from the semantic segmentation decoder, which also carries the features extracted at higher resolutions thanks to the skip connection discussed before.

3. The objective of leaf instance segmentation is to identify individual leaves in each plant. To achieve this, knowing the position of each distinct plant can provide more valuable context beyond the simple semantic information, which only indicates crops' locations and the high-resolution features of the input from the encoder. Thus, as we did before, we augment the skip

connections with the contribution from the plant instance decoder. Importantly, these skip connections also carry information from the semantic decoder and the network's encoder that are fed into the plant instance segmentation decoder.

This newly proposed skip connection scheme directly exploits the underlying hierarchy between the tasks, designed to realize a meaningful transfer of features from one branch, either encoder or decoder, to the following one. Extensive experiments reported in Sec. 3.2.2 suggest that these task skip connections lead to superior performance.

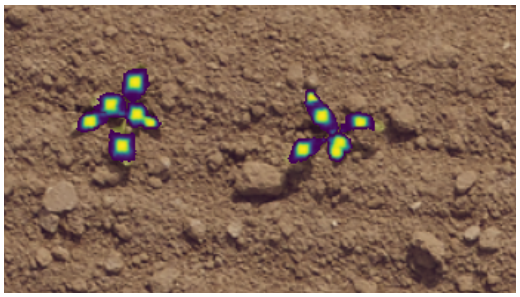
3.1.3 Post-Processing

Our automatic post-processing is a three-step procedure that exploits domain knowledge to handle occlusions and improve boundary detection in complex agricultural scenes. We show each step in detail for two exemplary plants in Fig. 3.4 for the leaf instance segmentation task. The same post-processing is applied to obtain the plant instances. The first step is inspired by Panoptic DeepLab [39] and aims to extract a single center for each object. Specifically, we take the center prediction from the decoder and filter it with the predicted semantic mask to discard any center that does not belong to the class of interest. Since the center prediction head commonly outputs blobs around the desired center, we perform a non-maximum suppression operation to reduce each blob to a single pixel. We show in Fig. 3.4 (a) the output of the prediction head and in Fig. 3.4 (b) the result after the non-maximum suppression, where each red dot is a predicted center.

Afterward, we need to assign each pixel to its center, which defines the individual instances. However, the offsets could point to regions of space close to more than one center. In the second step, we assign only those pixels whose offsets point to a single center. To achieve this, we construct an image of coordinates $\mathbf{I}_{\text{coord}}$, where each pixel $\mathbf{p}_{i,j}$ is a vector of values $(i, j), i \in \{1, ..., H\}, j \in \{1, ..., W\}$. We compute the Euclidean distance between $\mathbf{I}_{\text{coord}}$ and every ground truth center $\mathbf{c}$, producing for each center a ground truth distance map $\mathbf{D}^c \in \mathbb{R}^{H \times W}$. Then, we compute a predicted distance map $\hat{\mathbf{D}} \in \mathbb{R}^{H \times W}$ from the offsets, as

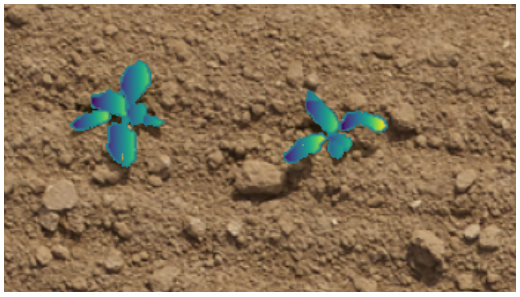
$$\hat{\mathbf{D}} = \sqrt{\hat{\mathbf{I}}^2_{\text{off},0} + \hat{\mathbf{I}}^2_{\text{off},1}}, \tag{3.6}$$

where, we remember, $\hat{\mathbf{I}}_{\text{off},0}$ is the image of the predicted offsets in the x-direction, and $\hat{\mathbf{I}}_{\text{off},0}$ is the image of the predicted offsets in the y-direction. We visualize in Fig. 3.4 (c) the offsets in the x-direction and in Fig. 3.4 (d) the offsets in the y-direction. We can see that each leaf has a gradient from yellow to dark purple, meaning that one extreme has high positive values and the other has high negative values. Finally, in Fig. 3.4 (e), we show the predicted $\hat{\mathbf{D}}$, where we

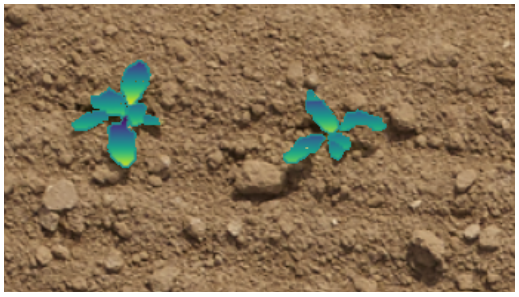


(a) Centers probabilities

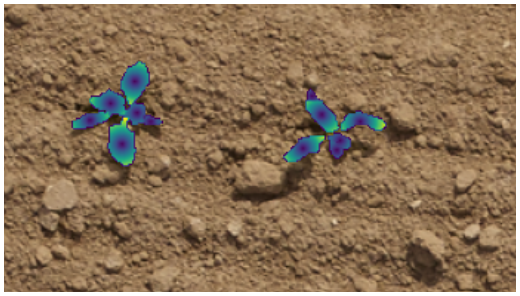
(b) Predicted centers



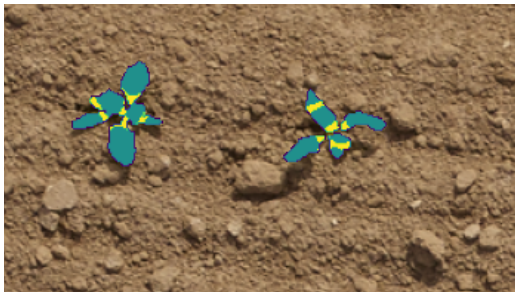
(c) $\hat{\mathbf{I}}_{\text{off},0}$



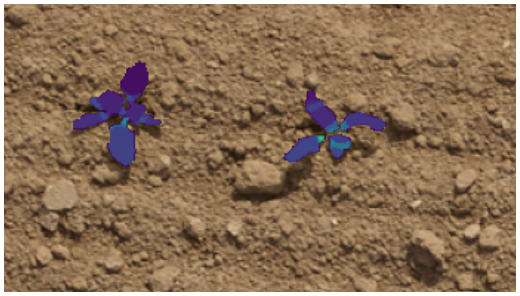
(d) $\hat{\mathbf{I}}_{\text{off},1}$



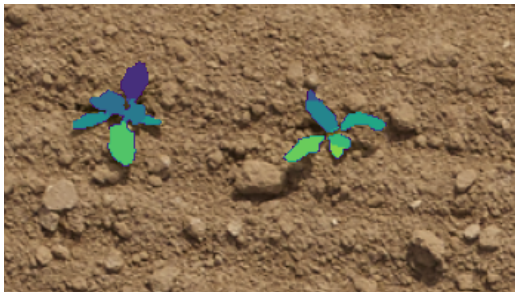
(e) $\hat{\mathbf{D}}$



(f) Assigned pixels



(g) Partial leaf instance



(h) Final leaf instance

Figure 3.4: Workflow of our post-processing for two exemplary plants. In (a) and (b) the predicted centers before and after non-maximum suppression. Then, we show the predicted offsets in the x (c) and y (d) directions, and the distance map $\hat{\mathbf{D}}$ computed from them in (e), where dark purple pixels have lower distance to the center they point to, while high distances are in yellow. In (f), we show in green the pixels assigned to an instance during step 2, and in (g) the partial leaf instance segmentation after step 2. In (h) the final result after assigning the remaining pixels in step 3.

can see that the distances change in a radial pattern from the leaves' centers. If the offsets point close to a center $\mathbf{c}$, we expect the predicted distance map to be similar to the ground truth one $\mathbf{D}^c$. Thus, defining a distance threshold τ, one pixel of coordinates (i, j) is assigned to the instance with center $\mathbf{c}$ if the term

$$\left|\left| D^c_{i,j} - \hat{D}_{i,j} \right|\right| \leq \tau, \tag{3.7}$$

holds for that instance only. In Fig. 3.4 (f), we see in green the pixels that have been assigned to an instance, and in yellow the pixels that have not been assigned yet. The instance segmentation at this point is displayed in Fig. 3.4 (g).

The third step takes care of the pixels that were not assigned to any instance. This can happen if their offset is pointing far from every extracted center or close to more than one. In this case, we apply a nearest-neighbor voting scheme. We compute the instance label that occurs the most between the $\mathrm{N}_{\mathrm{neigh}}$ closest neighbors and assign it to the current pixel. We show the final result of our post-processing operation in Fig. 3.4 (h). To enforce consistency between the masks, we filter all post-processing results with the semantic segmentation masks.

3.2 Experimental Evaluation

We present our experiments to show the capabilities of our method for joint semantic, plant instance, and leaf instance segmentation of RGB data. The results of our experiments confirm that our joint approach works on data collected in real fields, i.e., not in controlled environments; our novel scheme for the skip connections better exploits the hierarchical connections between the tasks improving the performance; and our post-processing achieves superior performance with respect to common state-of-the-art methods.

3.2.1 Experimental Setup

Datasets. We test our method on two RGB datasets: a sugar beets dataset introduced by Weyler et al. [222], denoted as SugarBeets, and GrowliFlower [106]. SugarBeets is composed of 1,316 images with a resolution of $512\,\mathrm{px} \times 1024\,\mathrm{px}$. The images are recorded by a UAV equipped with a PhaseOne iXM-100 camera mounted in nadir view. GrowliFlower is a dataset of cauliflower images. It is composed of 2,198 images with a resolution of $368\,\mathrm{px} \times 448\,\mathrm{px}$. The images are recorded by a UAV equipped with a Sony A7 rIII RGB camera and a MicaSense 5CH for multispectral image data. Both datasets provide an official data split that we adopt.

Metrics. We evaluate the performance of semantic segmentation computing the intersection over union (IoU) [60] of the "crop" class. The IoU provides a

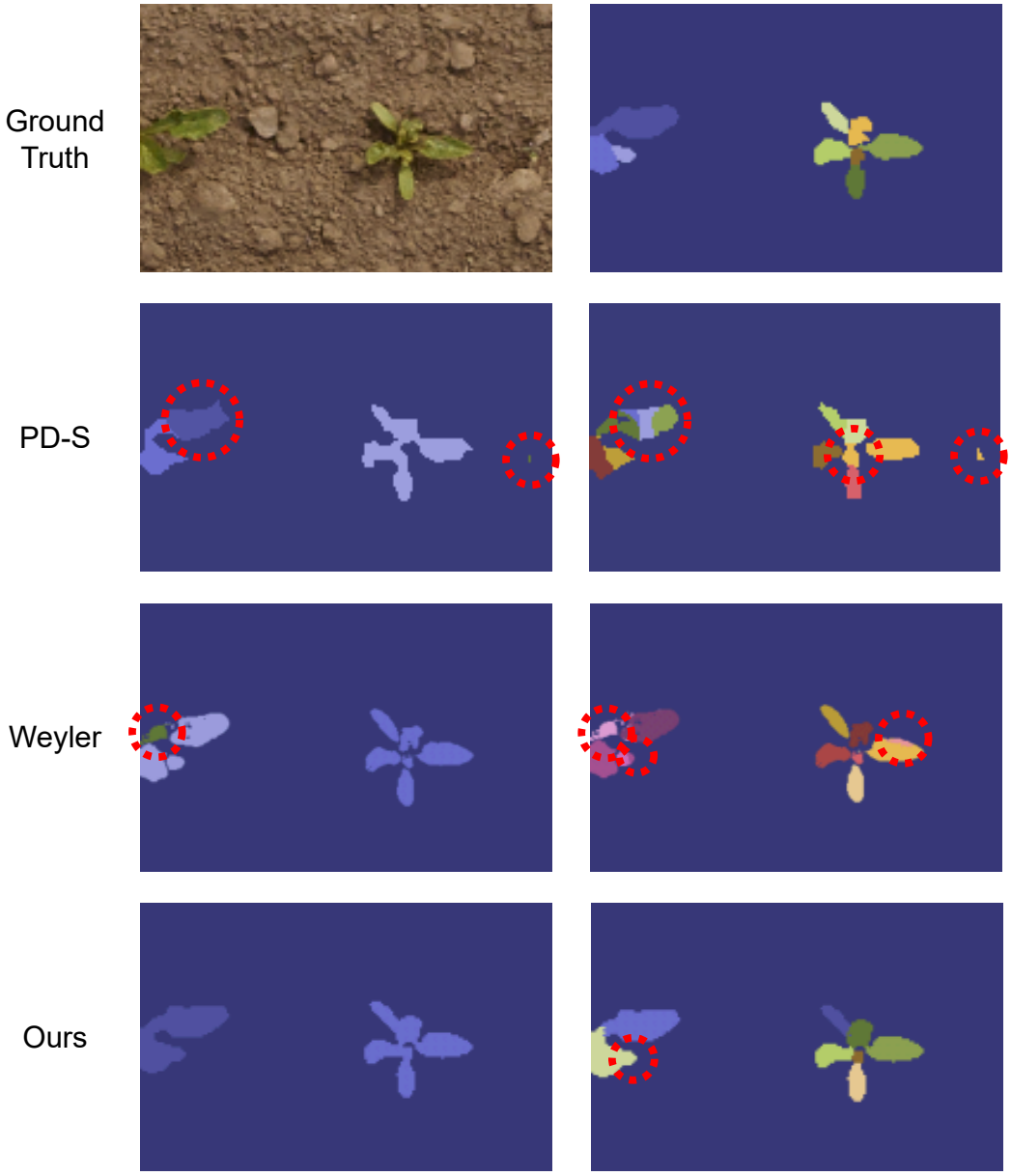


Figure 3.5: Qualitative results on the SugarBeets dataset for plant (left) and leaf (right) instance segmentation. We show the input image and ground truth leaf instances in the first row, and results from PD-S (second row), Weyler (third row), and our approach (last row). We highlight the prediction errors using red circles.

quantitative measure of the overlap between two regions: the pixels PR predicted as a given class or object by the network, and the pixels GT that belongs to that class or object in the ground truth labels. Formally, we compute the IoU as

$$\mathrm{IoU}\left(\mathrm{PR}, \mathrm{GT}\right) = \frac{|\mathrm{PR} \cap \mathrm{GT}|}{|\mathrm{PR} \cup \mathrm{GT}|}, \tag{3.8}$$

where $|\mathrm{PR} \cap \mathrm{GT}|$ is the number of true positives (TP), i.e., pixels correctly predicted, and $|\mathrm{PR} \cup \mathrm{GT}|$ is the number of pixels belonging to the class or object in either set. A perfect prediction yields an IoU of 1, indicating complete overlap between the sets. On the contrary, an IoU of 0 indicates no overlap at all. In practice, IoU is often multiplied by 100 and expressed as a percentage. The metric penalizes both false positives (FP) and false negatives (FN), providing a balanced assessment of the segmentation accuracy. False positives refer to pixels incorrectly predicted as part of the class or object when they are not, while false negatives are pixels that belong to the class or object in the ground truth labels but are not predicted as such.

For the plant and leaf instance segmentation, we evaluate our method with the panoptic quality (PQ) [109]. PQ combines semantic segmentation and instance segmentation, offering an evaluation of how well objects are detected and segmented. Given a set of predicted instances $\mathrm{pi}_i \in \mathrm{PI}$ and ground truth instances $\mathrm{gti}_i \in \mathrm{GTI}$, where each instance is represented by its set of pixels and its semantic class, the PQ is defined as

$$\mathrm{PQ} = \frac{\sum_{(\mathrm{pi},\mathrm{gti})\in \mathrm{TP}} \mathrm{IoU}\left(\mathrm{pi}, \mathrm{gti}\right)}{|\mathrm{TP}| + \frac{1}{2}|\mathrm{FP}| + \frac{1}{2}|\mathrm{FN}|}, \tag{3.9}$$

where a pair $(\mathrm{pi}, \mathrm{gti})$ belongs to the true positives (TP) only if the IoU exceeds a threshold, typically 0.5. Here, FP (false positives) are predicted instances with no matching ground truth, and FN (false negatives) are ground truth instances not matched by any prediction. Each predicted or ground truth instance can participate in at most one match. A high PQ indicates that the method successfully detects most object instances with precise localization and correct class labels. In contrast, a low PQ reflects either poor segmentation quality, incorrect classification, missed objects, or a combination of these factors. As already mentioned for the IoU, PQ is also often multiplied by 100 and expressed as a percentage.

Training details and hyperparameters. We use AdamW [130] without weight decay in all experiments. We set an initial learning rate of $5 \cdot 10^{-4}$ for the encoder and the semantic decoder, and $8 \cdot 10^{-4}$ for the instance decoders. We use a step scheduler for the encoder, with a step size of 25 epochs and $\gamma = 0.9$. For the instance decoders, we use an exponential scheduler with $\gamma = 0.99$. We train for 500 epochs. We initialize our network with the Xavier initialization [67]. The batch size is set to 1. We resize images from the SugarBeets dataset to

256 px×512 px to keep the aspect ratio. No resizing is applied to the GrowliFlower dataset. Additionally, we set $w_1 = 1$, $w_2 = w_3 = 0.1$, and $w_4 = w_5 = 50$ in Eq. (3.5), while in the post-processing we use $\mathrm{N}_{\mathrm{neigh}} = 5$ number of neighbors, grouping threshold $\tau = 6$ for the plant instance segmentation and $\tau = 2$ for the leaf instance segmentation in Eq. (3.7). We tuned all hyperparameters on the validation sets.

Baselines. We benchmark our model against commonly used segmentation approaches in the agricultural domain, including Mask R-CNN [83], denoted as MR, and Panoptic DeepLab [39], denoted as PD, which is a state-of-the-art model for panoptic segmentation. We use three variants of Panoptic Deeplab with different backbones: a small model that uses MobileNetV2 [191], called PD-S, a medium-size model with ResNet50 [84], called PD-M, and a big-size model with Xception65 [43], called PD-L.

All these baselines, however, can only address one instance segmentation task at a time and, thus, they need to be trained for either plants-only or leaves-only. We also compare with the work from Weyler et al. [222], denoted as Weyler, which addresses both, plant and leaf instance segmentation.

3.2.2 Experiments on Double Panoptic Segmentation

The first experiment evaluates the performance of our approach and its outcomes support the claim that we can jointly provide pixel-wise semantic, plant instance, and leaf instance segmentation. Tab. 3.1 and Tab. 3.2 show the IoU for the crops, and the panoptic quality for both plant (PQ_P) and leaf (PQ_L) instances on the SugarBeets and GrowliFlower datasets respectively. We also report the number of parameters of the networks and the end-to-end frame rate of each method at inference time (FPS).

Interestingly, the models that tackle all tasks simultaneously also have the fewest parameters, making them especially suitable for deployment on resource-constrained robotic systems. The additional parameters in our model compared to Weyler [222] are primarily due to the inclusion of the semantic segmentation decoder, which is our first output and filters the predictions of the subsequent decoders. Our approach is suitable for real-time operations with a frame rate that exceeds 20 Hz. Although some of the baselines performing only one instance task at a time (MR and all PD variants) achieve higher FPS, their frame rates cannot be directly compared to those of the joint models, as we would need to run two of such models to perform all tasks. Since most networks have similar FPS for the two tasks, this would generally double the needed time. However, on top of that, the approaches that need two models to perform all tasks would need an extra processing step to ensure consistency of the results increasing their final latency. The only baseline that addresses both, the plant and leaf instance

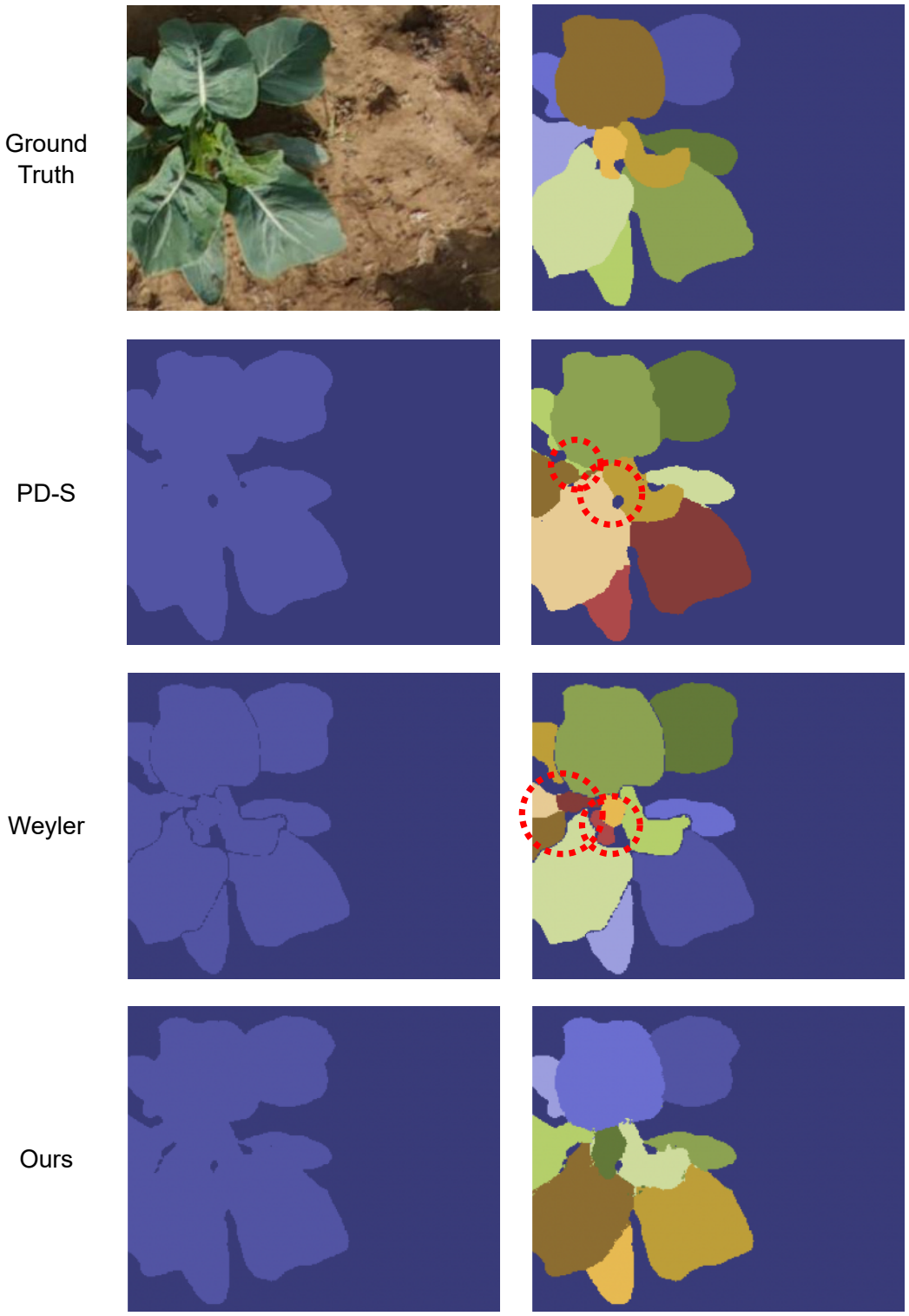


Figure 3.6: Qualitative results on the GrowliFlower dataset for plant (left) and leaf (right) instance segmentation. We show the input image and ground truth leaf instances in the first row, and results from PD-S (second row), Weyler (third row), and our approach (last row). We highlight the prediction errors using red circles.

Table 3.1: Performance of baselines and our model on the test set of the SugarBeets dataset. P and L stand for plant and leaf instance segmentation, respectively. Best results in bold. For the FPS we put in bold the best results for all three sections: semantic and plant instance segmentation, semantic and leaf instance segmentation, and joint semantic, plant, and leaf segmentation.

Model	P	L	IoU [%] ↑	PQ_P[%] ↑	PQ_L[%] ↑	#Params	FPS [s^{-1}] ↑
MR [83]	✓		46.2	47.8	-	43.9M	13.5
PD-S [39]	✓		75.4	69.4	-	7.7M	**93.5**
PD-M [39]	✓		75.5	69.8	-	55.3M	4.7
PD-L [39]	✓		76.4	71.1	-	69.6M	48.4
MR [83]		✓	64.9	-	53.6	43.9M	13.4
PD-S [39]		✓	75.4	-	50.8	7.7M	**93.7**
PD-M [39]		✓	76.7	-	54.4	55.3M	49.1
PD-L [39]		✓	76.3	-	52.9	69.6M	48.5
Weyler [222]	✓	✓	75.3	72.3	63.1	2.25M	0.14
Ours	✓	✓	**79.3**	**76.2**	**63.5**	2.4M	**26.3**

with one network is Weyler et al. [222], which is not suitable for real-time operations due to its relatively low framerate of 0.14 Hz on SugarBeets and 0.25 Hz on GrowliFlower. The bottleneck of the approach by Weyler et al. [222] is the post-processing based on the clustering of high-dimensional embeddings and the use of covariance matrices. While this shows to be effective, enhancing their performance compared to most other baselines, the clustering step is computationally demanding especially on large and high-resolution images.

In summary, our model with specifically designed skip connections and novel automatic post-processing operations significantly outperforms all baselines on every metric for the SugarBeets dataset. On GrowliFlower, our model achieves state-of-the-art results on both instance segmentation tasks, all while operating at the frame rate of common RGB cameras. Qualitative results and ground truth labels are shown in Fig. 3.5 for SugarBeets, and in Fig. 3.6 for GrowliFlower for PD-S [39], Weyler [222], and our approach.

3.2.3 Ablation on Residual Skip Connections

In this section, we provide additional experiments to evaluate how our novel residual skip connections scheme impacts the performance of the three tasks. We perform the experiment on the validation set of the SugarBeets dataset. We show the results in Tab. 3.3. In particular, we use the same network as the one we propose with no skip connections (A), typical encoder-decoder skip con-

Table 3.2: Performance of baselines and our model on the test set of the GrowliFlower dataset. P and L stand for plant and leaf instance segmentation, respectively. Best results in bold. For the FPS we put in bold the best results for all three sections: semantic and plant instance segmentation, semantic and leaf instance segmentation, and joint semantic, plant, and leaf segmentation.

Model	P	L	IoU [%] ↑	PQ_P[%] ↑	PQ_L[%] ↑	#Params	FPS [s^{-1}] ↑
MR [83]	✓		25.4	27.9	-	43.9M	9.6
PD-S [39]	✓		83.1	69.9	-	7.7M	43.4
PD-M [39]	✓		82.0	68.0	-	55.3M	**47.6**
PD-L [39]	✓		82.7	69.4	-	69.6M	23.8
MR [83]		✓	53.8	-	41.0	43.9M	16.2
PD-S [39]		✓	**84.4**	-	58.8	7.7M	**76.5**
PD-M [39]		✓	80.2	-	43.4	55.3M	41.6
PD-L [39]		✓	82.8	-	50.1	69.6M	30.3
Weyler [222]	✓	✓	65.8	67.8	69.4	2.25M	0.25
Ours	✓	✓	80.2	**89.2**	**71.0**	2.4M	**20.7**

Table 3.3: Comparison between different skip connections schemes, with or without gradient flow. Best results in bold.

	Skip Connections	Attached	IoU [%]	PQ_P [%]	PQ_L [%]
A	None		84.4	75.5	65.1
B	Encoder	✓	83.3	79.4	65.6
C	Task + Encoder		83.2	78.9	65.6
D	Task	✓	84.4	81.1	66.0
E	(Ours) Task + Encoder	✓	**84.5**	**81.7**	**67.8**

nections [186] (B), hierarchical skip connections with no gradient flow (C), skip connections without summing the contribution from the encoder (D). If we do not use any skip connection the panoptic qualities are noticeably lower, because the corresponding decoders have no help from previous features. Using encoder-decoder skip connections (B) improve the panoptic qualities thanks to the access to encoder features. However, since the encoder must capture features relevant to all tasks, this harms the semantic segmentation performance. Interestingly, we notice no improvement from the hierarchical skip connections with no gradient flow (C), where skip connections are detached and thus do not participate in the backward pass, since the feature flow does not play any role in the optimization, leading to suboptimal performance. On the other hand, hierarchical skip connections without the encoder contribution (D) substantially improve performance

Table 3.4: Comparison between different post-processing operation, starting from identical network's predictions. Best results in bold.

Post-processing	IoU [%]	PQ_P [%]	PQ_L [%]
Panoptic DeepLab [39]	**84.5**	79.0	47.2
Ours	**84.5**	**81.7**	**67.8**

compared to only using the skip connections from the encoder (B). This suggests that decoder-level features are more relevant than restoring features from the encoder when it comes to tasks that present an underlying hierarchical structure. Nevertheless, our skip connection scheme (E) is the one achieving the best result across all tasks.

3.2.4 Ablation on Post-Processing

We also evaluate the contribution of our novel post-processing in Tab. 3.4. We use our model as introduced in Sec. 3.1, trained with our loss function and skip connections scheme. Since the architecture and training are unchanged, the predictions of the network are identical between the two experiments. This allows us to compare the effectiveness of our post-processing operation with the state-of-the-art instance post-processing from Panoptic DeepLab. Since the post-processing of instances does not play any role in the semantic segmentation, the IoU remains the same. We can see that our post-processing leads to significant improvements in instance segmentation, especially when clustering leaves. We find that this improvement is due to the way our post-processing handles the offsets that point between multiple centers or to regions distant from any center. The post-processing from Panoptic DeepLab simply assigns the pixel to the center closest to the pixel pointed by the offset. This is particularly problematic during leaf instance segmentation, where several centers are spatially close. On such occasions, even a one-pixel shift in the offset can result in incorrect instance segmentation. We address these issues using the threshold τ in the second step, and the voting mechanism in the third step of our post-processing. This enables us to correct for ambiguous offsets that might otherwise lead to erroneous instance segmentation.

3.3 Discussion

Numerous works have addressed semantic, plant instance, or leaf instance segmentation independently, achieving admirable performance. However, such approaches do not exploit task hierarchy and, since the predictions come from independent models, their results are often inconsistent, requiring additional steps

to produce coherent predictions.

To address this limitation, we propose integrating semantic, plant instance, and leaf instance segmentation into a single unified model. Our approach progressively performs the tasks, capturing both high-level semantic understanding and fine-grained details. Organizing the model hierarchically improves both segmentation quality and boundary detection by leveraging the context gained from previous tasks, while also enabling extending the model to additional tasks such as fruit segmentation. A key component of our approach is the novel domain-specific post-processing operation, which enforces structural consistency and reduces over-segmenting the instances, especially in areas with dense vegetation or occlusion. The use of domain-specific thresholds and the refinement step for pixels with ambiguous offset predictions provides an effective way to incorporate prior agricultural knowledge into the segmentation pipeline, without requiring additional data or adaptation of the network.

Our experimental evaluation confirms that the hierarchical structure of our method improves the performance of all tasks while producing consistent predictions. Notably, this does not come at the cost of a significantly larger model, which could be problematic for deployment on resource-constrained robotic platforms, nor does it slow inference times that could hinder real-time operability. Even without achieving the highest IoU on the GrowliFlower dataset, our method has superior panoptic quality on both plant and leaf instances. This indicates that even with slightly lower semantic segmentation performance, the overall instance-level understanding is significantly improved. The small gap in IoU performance, which is at most a drop of 4.2 percentage points, can be attributed to our compact architecture, which jointly addresses three tasks with fewer parameters. Despite this, we observe a notable minimum gain of +19.3 percentage points in PQ for plants and +12.2 percentage points in PQ for leaves when compared to models with higher IoUs. Moreover, Tab. 3.4 shows that our novel post-processing strategy contributes to the improvements by enhancing final predictions even when the raw network outputs remain unchanged, demonstrating the effectiveness of using in-domain knowledge to obtain more accurate predictions.

Relying on the predictions of previous tasks to guide the subsequent ones can be a risk, especially when the network is applied to out-of-domain data. It is well established in the literature that fully supervised methods tend to perform poorly in unfamiliar scenarios [66, 232]. In such cases, an inaccurate semantic segmentation prediction would be propagated to the following tasks, making it difficult for the system to recover and achieve satisfactory performance. To address this without re-training the network, one could design a mechanism that adaptively weights the influence of previous task predictions based on their estimated reliability. Similarly, the hand-tuned thresholds used in the post-processing need to

be adapted to new image resolutions or crop species, limiting the applicability of the approach to new scenarios. Addressing this issue is not trivial because the quality of instances is quantified by the PQ, which is not differentiable and thus cannot be directly optimized. One possible way would be to compute the instance mask using the predicted thresholds and optimize the threshold via an auxiliary loss. One solution is computing the IoU between the masks obtained for each instance using the predicted threshold and the ground truth masks, taking into account the difference between the number of predicted and ground truth instances as formulated in the work by Cheng et al. [42].

3.4 Conclusion

In this chapter, we introduced a novel approach for joint hierarchical semantic, plant instance, and leaf instance segmentation of RGB images. Our method leverages the task hierarchy via a newly designed skip connection scheme, enabling each task to support the following ones in a structured and efficient manner. In addition, we introduced a novel post-processing strategy that significantly boosts instance segmentation performance by incorporating structural knowledge of the agricultural environment by means of the threshold τ. This threshold depends on the morphology of the investigated crops, making it a domain-specific parameter. The experiments show improved segmentation accuracy and demonstrate the importance of domain knowledge in training models to surpass the capabilities of purely data-driven methods. This is shown in our last ablation study, where we isolate the contribution of our novel post-processing operation. Applying the two different post-processings on identical network predictions, we can see the improvement obtained only thanks to our post-processing. The experiments indicate that exploiting the task hierarchy and integrating domain knowledge are both crucial for effective scene understanding in agricultural fields. In the next chapter, build on these findings to extend the idea of exploiting domain knowledge for improving model initialization and learning across various perception tasks, reducing reliance on manual annotations and boosting generalization.

Chapter 4

Exploiting Domain Knowledge for Task-Agnostic Pre-Training

Deep learning approaches, such as the one previously introduced in this thesis, have significantly improved the capabilities of robotic systems in agriculture, particularly in tasks requiring precise scene understanding. In Chapter 3, we proposed a fully supervised method that jointly performs semantic, plant instance, and leaf instance segmentation using RGB images. Such tasks enable automatic assessment of plant health [76, 138], identification and localization of weeds [223], detection of plant diseases [73], and monitoring of the growing conditions in the field [225]. However, like most supervised methods, it needs a large amount of labeled data to achieve satisfactory performance. Acquiring labeled data is expensive and time-consuming. Moreover, for agricultural data, the need for expert knowledge to produce high-quality and accurate annotations poses an additional challenge.

To address this limitation, some researchers use semi-supervised approaches to reduce the need for per-pixel labeled images [134] or leverage background knowledge [150]. Recent work in self-supervised pre-training showed promising results, where we can pre-train a network without relying on supervision by manual labels. Pre-training on the ImageNet dataset [52] is a common way to reduce the number of training samples and the training time needed by the network to converge. Several other methods use a contrastive loss and strong augmentation techniques [28,37,71,82,238] to extract meaningful representations of the images, which are called embeddings and are high-dimensional vectors. Embeddings produced in such a way are robust to the augmentations applied at training time, focusing on more relevant details of the images. However, the applied data augmentations need to be selected carefully so as not to lose features useful for the semantic understanding of the image. In this chapter, we study self-supervised representation learning to improve the perception of agricultural robots. Fig. 4.1

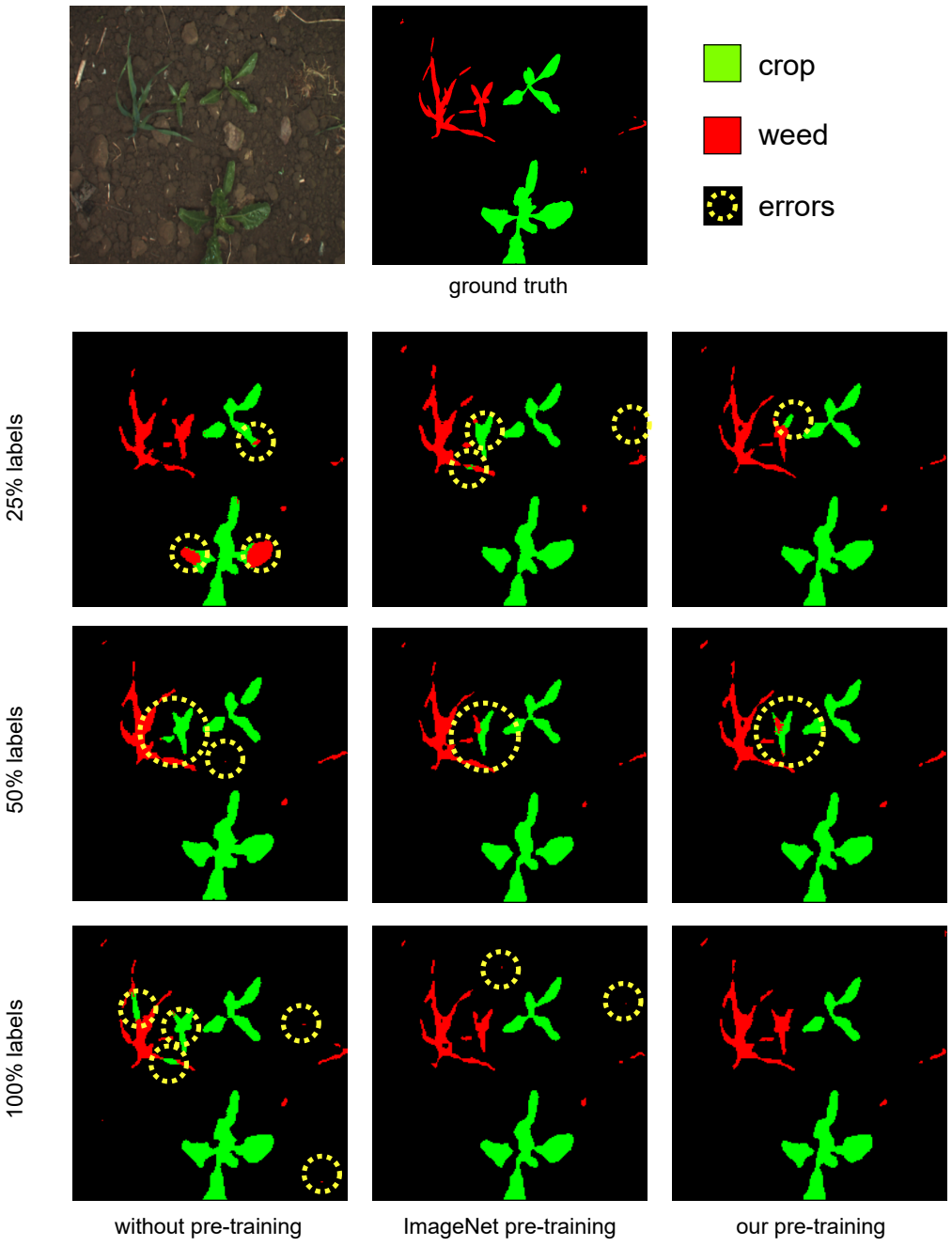


Figure 4.1: Results on semantic segmentation with different amounts of data and pre-trainings. We achieve better or comparable results with $\frac{1}{4}$ of the epochs and $\frac{1}{1000}$ of the images. The dotted circles highlight the errors in the results.

illustrates how the pre-training strategy affect the semantic segmentation results and shows higher accuracy and fewer errors for our approach.

The main contribution of this chapter is a pre-training strategy for the plant domain, which will reduce the number of labeled images needed, and a newly defined augmentation policy. We study existing augmentations and propose domain-specific ones to boost performance on different downstream tasks, i.e., the final task that we want the network to learn after pre-training, such as semantic or instance segmentation. We investigate how self-supervised pre-training on domain-specific data leads to better models that can learn with less labeled data on multiple agricultural perception tasks. Specifically, in this chapter, we present results on the semantic segmentation and leaf instance segmentation tasks and show that our pre-training improves the performance with respect to other state-of-the-art methods on both tasks. Our experiments suggest that using domain-specific pre-training can further reduce the number of labeled images needed, and confirm that the augmentation policy needs to be domain-specific, taking into account the order and design of the augmentations.

4.1 Our Approach for Effective Agricultural Pre-Training

We aim to learn an abstract representation that will serve as a starting point for further learning tasks in the domain of the perception of plants. By deploying robots in the fields, we can quite easily collect a large amount of *unlabeled* data. This offers the potential to build systems to train networks in a self-supervised fashion. For our perception task, we pre-train the network encoder following Barlow Twins proposed by Zbontar et al. [238]. Most contrastive learning approaches use positive and negative pairs, encouraging the network to produce similar representations for the positive pairs and different representations for the negative pairs. Instead, we decided to use Barlow Twins, which does not use negative pairs. This was a major advantage for us, since in the agricultural domain most images are very similar, all depicting plants in the fields. The pre-training approach, as well as the architecture for the tasks, is not our main focus. Our contribution lies in a novel domain-specific augmentation policy to boost the performance of the final system. We evaluate our pre-training on semantic and leaf instance segmentation.

4.1.1 Barlow Twins

Barlow Twins learns representations in a self-supervised fashion via redundancy reduction. Here, we briefly summarize its relevant parts and refer to the original

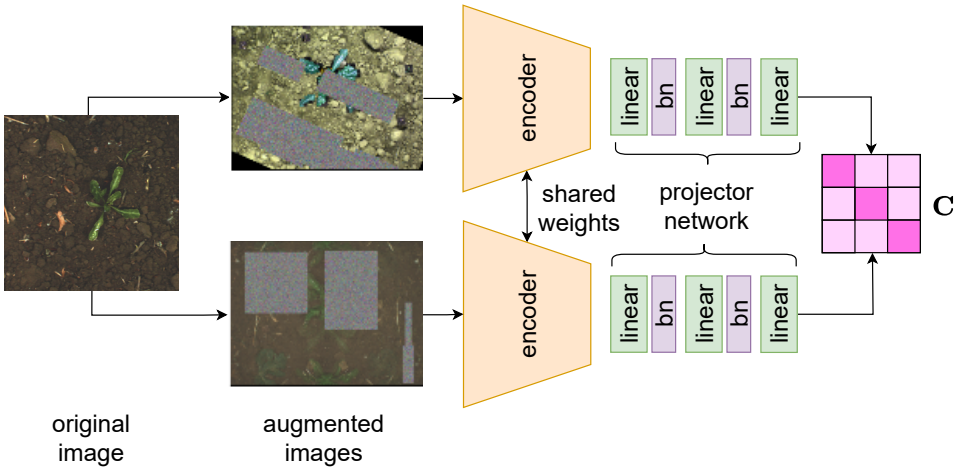


Figure 4.2: Overview of our architecture. For each image, we build two views using different augmentations and feed them into the network to produce their embeddings. We build the cross-correlation matrix $\mathbf{C}$ from the two embeddings and then compute the loss, that encourages $\mathbf{C}$ to be an identity matrix. In this way, we induce the network to produce non-redundant embeddings that are as close as possible for different views of the same input image.

paper [238] for more details. Barlow Twins uses a siamese network with shared weights, as depicted in Fig. 4.2. The two inputs are two different views computed from the same input image applying different augmentations. The encoder is a ResNet50 [84] without the final classification layer, followed by a projector network. The projector network has two identical blocks – linear layer, batch normalization, and rectified linear units – followed by one linear layer.

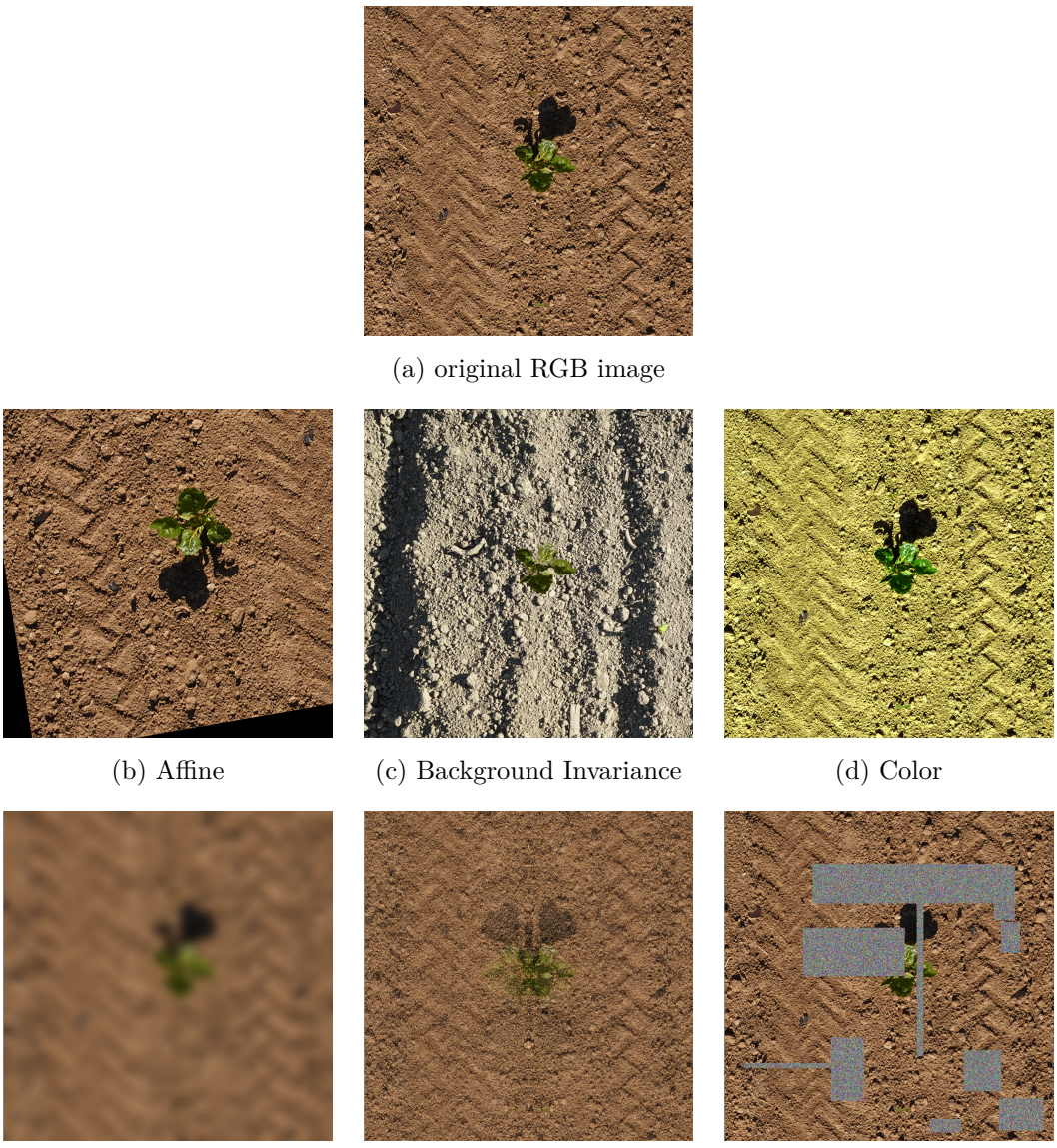
Zbontar et al. [238] build for each input image $\mathbf{I}$ two augmented views $\mathbf{I}_1$, $\mathbf{I}_2$ that are fed into the network to produce two distinct embeddings $\mathbf{z}_1$, $\mathbf{z}_2 \in \mathbb{R}^{\mathrm{D}}$. They compute the loss directly on $\mathbf{z}_1$ and $\mathbf{z}_2$. The first step is to construct the cross-correlation squared matrix $\mathbf{C} \in \mathbb{R}^{\mathrm{D}\times\mathrm{D}}$ from the embeddings normalized over the batch dimension. This matrix has values between -1 (anti-correlation) and 1 (correlation). Then, the loss is:

$$\mathcal{L}_{\mathrm{BT}} = \sum_i (1 - \mathbf{C}_{ii})^2 + \lambda \sum_i \sum_{j \neq i} \mathbf{C}_{ij}^2 \ , \tag{4.1}$$

where λ is a weight to trade-off two parts: the invariance term, which encourages the diagonal elements of $\mathbf{C}$ to be 1, and the redundancy reduction term, which penalizes the non-zero off-diagonal elements of $\mathbf{C}$.

4.1.2 Augmentations

Augmentations play a fundamental role in self-supervised learning. The stronger they are, the more the network focuses on relevant and stable features to represent



(a) original RGB image

(b) Affine

(c) Background Invariance

(d) Color

(e) Gaussian Blur

(f) Mix

(g) Random Erasing

Figure 4.3: We applied all of our augmentations to a single image to show one possible outcome for each one. In our pre-training strategy they are applied sequentially producing even more variations of the input.

the images. It has been recently shown [49], that augmentations effectively act as implicit kernel modifications within the network. However, augmentations are easier to develop and more practical to use, being architecture agnostic. We use our augmentation policy as common in the literature [71] plus domain-specific knowledge. In Fig. 4.3, we show the result of each augmentation applied to one sample image for illustration purposes. We now present all of the augmentations we use, first the general-purpose transformations, i.e., affine, gaussian blur, and random erasing, and then the domain-specific ones, i.e., color jittering, mixing, and background invariance.

4.1.2.1 Affine Transformation

The affine transformation, $\mathbf{T}_{\text{affine}} \in \mathbb{R}^{3\times3}$ rotates, translates, scales, and shears the input image. It makes the network invariant to such transformations which are common when working with robots of different sizes and cameras. More specifically, $\mathbf{T}_{\text{affine}}$ is given by

$$\mathbf{T}_{\text{affine}} = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} , \tag{4.2}$$

where $\mathbf{A} \in \mathbb{R}^{2\times2}$ contains an isotropic scaling factor in $[0.5, 2]$, a rotation in $[-\pi, \pi]$ and the shearing along the two axes randomly sampled in $[0.25, 0.75]$, while $\mathbf{t} \in \mathbb{R}^2$ is a translation vector with each component $t_x, t_y \in [-0.25 \cdot W, 0.25 \cdot H]$, where W and H are the image width and height respectively.

4.1.2.2 Gaussian Blur

We blur the image using a random standard deviation $\sigma_{\text{gb}} \in [0.1, 2]$. The purpose of this augmentation is to help the network focus on the image structure across different scales and resolutions.

4.1.2.3 Random Erasing

Random erasing [244] selects multiple rectangles inside of the image and substitutes the pixels' values with random values in $[0, 255]$. Given the minimum percentage of the image that has to be removed, it picks rectangles of different sizes and aspect ratios until the deleted area is at least the minimum desired area. We slightly change the implementation to enforce the use of multiple rectangles rather than a single large one. We use this augmentation to make the network less sensitive to occlusions and shadows.

4.1.2.4 Color Jittering

Color jittering changes the brightness, contrast, hue, and saturation of the image. Instead of the symmetrical range of values for the hue $(-0.1, 0.1)$ from the literature, we use $(0, 0.125)$ as the range. The commonly used symmetrical range can produce a wider range of colors, but some of the augmented images will not look realistic. This is usually not a problem when pre-training for classification for large-scale problems, where color is not the main feature of the object. However, in the agricultural domain, knowing a realistic range of colors for the plants is crucial to distinguish the vegetation from the soil and to identify ill or damaged plants, where color is often the major discriminator [208].

4.1.2.5 Mixing

Zhang et al. [241] propose to mix two different images into one via linear interpolation, we instead use a single image $\mathbf{I}$. We create two copies of $\mathbf{I}$, one flipped on the x-axis called $\mathbf{I}_x$ and one on the y-axis called $\mathbf{I}_y$. We sample each pixel in the augmented image from $\mathbf{I}$, $\mathbf{I}_x$, or $\mathbf{I}_y$ using a uniform probability over the three images. This can simulate motion due to the wind or water uptake, or holes eaten by insects.

4.1.2.6 Background Invariance

This transformation cuts plants from the current image and pastes them into a different soil background. This is important to expose the network to different combinations of crops and soils. Specifically, we perform the following steps:

1. Compute a normalized image as

$$\mathbf{I}_{\text{norm}}(i,j) = \frac{\mathbf{I}(i,j) - \boldsymbol{\mu}_{\mathbf{I}}}{\boldsymbol{\sigma}_{\mathbf{I}} + \epsilon} \ , \tag{4.3}$$

   where i, j are pixel coordinates, $\boldsymbol{\mu}_{\mathbf{I}} \in \mathbb{R}^3$ and $\boldsymbol{\sigma}_{\mathbf{I}} \in \mathbb{R}^3$ are mean and standard deviation of $\mathbf{I}$, and $\epsilon = 10^{-8}$.

2. Compute the vegetation mask $\mathbf{M}$ following Woebbecke et al. [227]. Specifically, $\mathbf{M}$ is given by

$$\mathbf{M} = 2\mathbf{I}_{\text{G}} - \mathbf{I}_{\text{R}} - \mathbf{I}_{\text{B}} \ , \tag{4.4}$$

   where $\mathbf{I}_{\text{R}}$,$\mathbf{I}_{\text{G}}$ and $\mathbf{I}_{\text{B}}$ are the color channels of $\mathbf{I}_{\text{norm}}$.

3. Convert $\mathbf{M}$ to a binary mask using a threshold τ_{M}, i.e., all pixels above τ_{M} are set to 1, the others are set to 0.

4. Refine $\mathbf{M}$ using 2 rounds of erosion with kernel size $(2, 2)$, 4 rounds of dilation with kernel size $(6, 6)$.

5. Cut the vegetation and paste it at a random location on a random soil image from a dataset of images whose vegetation mask **M** is below a given threshold, i.e., less than 5% of the image.

All the augmentations are applied with a certain probability, tuned from the results in Sec. 4.2.4.

4.1.3 Downstream Tasks

In an application, the pre-trained models are fine-tuned on specific downstream tasks. Our approach is network-independent since it is mainly a strategy to provide a better network initialization. Thus, we pre-train ResNet50 [84] for semantic segmentation and an ERFNet [184] encoder for leaf instance segmentation. For both tasks, we use ERFNet-like decoders, as those used in the network proposed in Chapter 3 for all tasks. The ResNet50 [84] architecture has been chosen for the semantic segmentation task in order to use publicly available initialization weights and, thus, provide a fair comparison against state-of-the-art methods.

4.1.3.1 Semantic Segmentation

Semantic segmentation predicts a class for each pixel of the image, in our application example, crop, weed, and soil. The decoder outputs an image $H \times W \times K$, where H and W are the height and width of the input image, and K is the number of semantic classes. Instead of connecting the decoder at the end of the ResNet50 [84], we discard the last two layers to preserve more spatial information. The truncated backbone can still be initialized with the pre-trained weights without changing anything. Following Rahman et al. [174], we directly optimize the Intersection over Union (IoU) metric during training.

4.1.3.2 Leaf Instance Segmentation

Leaf instance segmentation predicts a pixel-wise mask for each leaf. This task allows discovering the shape and size of individual leaves, and counting them, which is fundamental in determining the growth stage of the plant [61]. We use the network and loss proposed by Weyler et al. [222], which we also used as a baseline in Chapter 3. This setup demonstrates that our pre-training approach is network-independent and can be applied without relying on prior components from this thesis, making it broadly usable without loss of generality. One decoder predicts the center locations of each leaf, the other the offsets pointing at the specific leaf and plant center plus clustering parameters for their post-processing. We feed the predicted and ground truth masks to the Lovász Hinge Loss [17]. For more details, refer to the original paper by Weyler et al. [222].

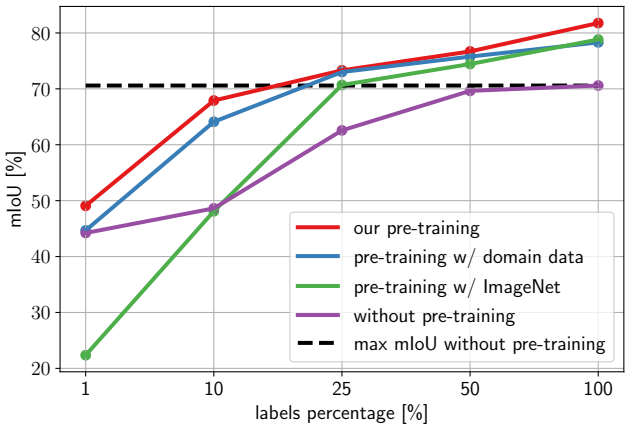


Figure 4.4: Comparison of the mIoU with different amounts of labels after fine-tuning for 100 epochs (semantic segmentation). The number of images for each label percentage is: 14 for 1%, 140 for 10%, 362 for 25%, 724 for 50%, and 1,450 for 100%.

4.2 Experimental Evaluation

The focus of this work is showing that our domain-specific self-supervised pre-training together with augmentation policy has the potential to perform better than the commonly used supervised pre-training on ImageNet. We show this for images from the agricultural robotics domain. In our experimental evaluation, we show that even just pre-training on plant images, i.e., in-domain data, improves the performance of the investigated downstream tasks. Furthermore, our results suggest that using domain-specific pre-training further reduces the need for labels and that considering the design of each augmentation is crucial to apply them in the best order.

4.2.1 Experimental Setup

Datasets. We pre-train on 18,000 unlabeled images of sugar beets from four different locations (Ancona in Italy, Bonn and Stuttgart in Germany, and Eschlikon in Switzerland) [32]. For the semantic segmentation task, we use the official annotated split from the same dataset [32] that contains 2,148 images: 1,450 for training, 478 for validation, and 220 for testing. For the leaf instance segmentation, we use the SugarBeets dataset introduced in Chapter 3, with 1,316 images; 746 for training, 292 for validation, and 278 for testing. We pre-train on a single NVIDIA RTX A6000 GPU and fine-tune on a single Quadro RTX 5000 GPU.

Metrics. We evaluate the results on semantic segmentation using the mean IoU (mIoU) [60], which is the mean of the IoU computed over all $K = 3$ semantic classes, i.e., soil, crop, and weed. For the leaf instance segmentation task, we report the average precision (AP) and recall (AR), and the absolute difference

Table 4.1: Comparison of average precision (AP) and recall (AR) on plants (p) and leaves (l) for the three main approaches. The number of images for each label percentage is: 7 for 1%, 74 for 10%, 186 for 25%, 373 for 50%, and 746 for 100%.

Percentage of Labels	Pre-Training	AP_p [%]	AR_p [%]	AP_l [%]	AR_l [%]
100%	none	54.3	60.5	48.7	68.3
	ImageNet	55.1	61.2	59.7	68.9
	ours	**55.6**	**62.9**	**64.4**	**69.2**
50%	none	50.3	59.0	45.6	60.1
	ImageNet	52.4	60.1	52.7	61.5
	ours	**54.6**	**60.8**	**54.6**	**62.7**
25%	none	48.0	55.2	42.0	46.1
	ImageNet	50.2	56.1	50.6	56.6
	ours	**50.9**	**56.9**	**53.8**	**60.6**
10%	none	46.6	54.0	20.7	39.6
	ImageNet	46.8	53.7	29.5	38.4
	ours	**48.0**	**54.2**	**42.5**	**49.2**
1%	none	0.0	0.0	0.1	0.3
	ImageNet	0.0	0.0	0.4	0.2
	ours	**1.1**	**8.3**	**0.9**	**5.6**

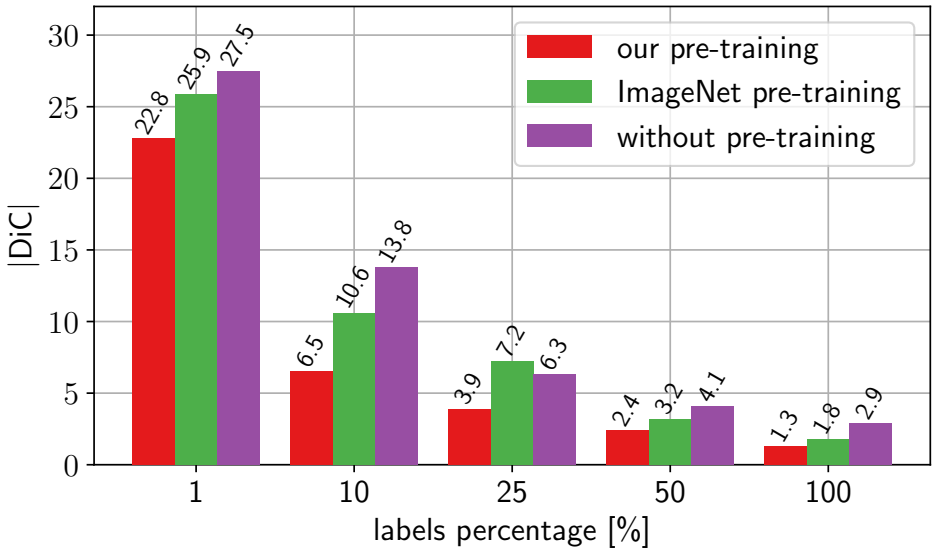


Figure 4.5: The average |DiC| for the three approaches with increasing number of labeled images. The lower the better.

in count (|DiC|) of the leaves. AP evaluates the precision of the instances across various IoU thresholds τ_i, where i goes from 0.5 to 0.95 in steps of size 0.05, as

$$\mathrm{AP} = \frac{1}{K} \sum_i \frac{\mathrm{TP}\left(\tau_i\right)}{\mathrm{TP}\left(\tau_i\right) + \mathrm{FP}\left(\tau_i\right)}, \tag{4.5}$$

where $\mathrm{TP}\left(\tau_i\right)$ and $\mathrm{FP}\left(\tau_i\right)$ are the number of true positives and false positives when threshold τ_i is applied.

AR measures the fraction of ground truth instances that are correctly detected. For a given threshold τ_i, we compute it as

$$\mathrm{AR} = \frac{1}{K} \sum_i \frac{\mathrm{TP}\left(\tau_i\right)}{\mathrm{TP}\left(\tau_i\right) + \mathrm{FN}\left(\tau_i\right)}, \tag{4.6}$$

where $\mathrm{FN}\left(\tau_i\right)$ is the number of false negatives when threshold τ_i is applied.

Training details and baselines. We pre-train our encoder for 250 epochs with batch size 128. We use AdamW [130] with a fixed learning rate of $2 \cdot 10^{-4}$, and a weight decay of 10^{-6}. We compare our method with the results obtained using the same network without any pre-training, i.e., randomly initialized, and pre-trained in a supervised fashion on the general-purpose ImageNet dataset [52].

4.2.2 Our Pre-training vs. Non-specific Pre-training

The first experiment analyzes how self-supervised domain-specific pre-training provides a more effective initialization while decreasing the need for labeled im-

ages, time, and computational resources. We fine-tune our model and the model pre-trained on ImageNet with different amounts of labels.

Semantic Segmentation. Fig. 4.4 suggests that when using a sufficient number of labels, different pre-training strategies perform similarly. The fewer labels we use, the wider the gap. The ImageNet pre-training requires more labeled data to adjust to the agriculture domain. Our pre-training performs better requiring less data for pre-training i.e. 18,000 images against the 1,281,167 from ImageNet, and epochs i.e. 250 against 1,000. For this experiment only, we also pre-train on domain-specific data with the augmentations from the literature. The results confirm that domain-specific augmentations are a key component to obtain the best performance.

Leaf Instance Segmentation. Tab. 4.1 confirms the utility of pre-training on domain-specific data to boost the performance and reduce the number of labeled images needed. Our pre-training boosts every metric in every scenario. In Fig. 4.5, the difference in count shows a similar trend. When using less than 10 images, none of the approaches can properly segment the leaves. Using 74 images (10%), our pre-training can already reduce the uncounted leaves to ≈ 6 per image (each image can have between 2 and 5 plants in it). We illustrate qualitative results in Fig. 4.6, where we highlight with yellow dotted circles the leaf instance segmentation errors. These results confirm that our pre-training reduces incorrect predictions even for fine-grained tasks.

4.2.3 Pre-training vs. No Pre-training

This experiment aims to confirm that using our pre-trained backbone boosts the final performance and reduces the reliance on labeled data compared to training the model without any pre-training.

Semantic Segmentation. In Fig. 4.4, we see how pre-training boosts the performance when using the same routine and number of labeled images. Our pre-trained model performs better using up to 25% of the annotated data. With less than 10% of the labeled data, pre-training on ImageNet is hurting the performance. The reason may be that the network expects to see objects from the ImageNet dataset distribution and requires the labeled data to adapt to the agricultural domain. Our pre-training does not suffer from this issue.

Leaf Instance Segmentation. Training the randomly initialized network with only a few labeled images rapidly deteriorates the performance. Fig. 4.5 and Tab. 4.1 show that we can achieve the same performance using half of the data. Our pre-training boosts all metrics, and it produces instance masks using only 7 images (1%). The qualitative results in Fig. 4.6 demonstrate that pre-training enhances the network's ability to distinguish close instances.

(a) without pre-training

(b) with ImageNet pre-training

(c) with our pre-training

(d) Ground Truth labels

Figure 4.6: Qualitative results of the leaf instance segmentation task, where each color corresponds to a different instance. All results are obtain after fine-tuning the network with 50% of the labels. We show the results when the network is randomly initialized with no pre-training (a), when it is initialized with the ImageNet pre-training (b), and with our pre-training (c). In (d), we show the ground truth labels. We highlight the mistakes in yellow dotted circles.

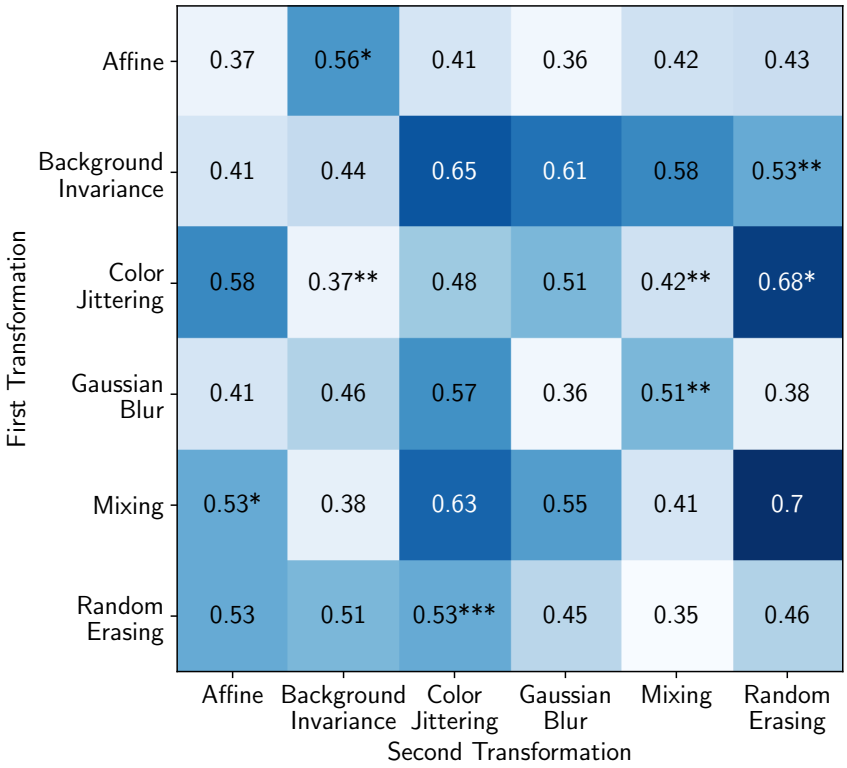


Figure 4.7: The mIoU for different combinations of transformations. We fine-tune with minimum 100 epochs or until convergence. We use different number of asterisks to indicate how many more epochs were needed to converge, where (*) means 10 extra epochs; (**) 40 extra epochs; and (***) 100 extra epochs.

Table 4.2: The mIoU [%] after fine-tuning (100 epochs on semantic segmentation) with 20, 40, 60, 80, and 100 epochs of pre-training using only the color transformation.

Color	**pre-training epochs**				
Augmentation	**20**	**40**	**60**	**80**	**100**
standard	**23.44**	**23.77**	23.89	19.92	13.89
Ours	17.61	21.11	**31.28**	**32.52**	**42.80**

4.2.4 Relevance and Order of Augmentations

We use a shorter training routine to analyze which augmentations work better in the agricultural domain. We pre-train each combination for 50 epochs on a subset of the pre-training dataset and then fine-tune on a subset of the labeled dataset for semantic segmentation. Chen et al. [37] did a similar experiment on a general-purpose setting and showed results similar to ours.

In Fig. 4.7, we see that changing the order of the augmentations impacts both the mIoU and the training time. The combinations that took more time are those that make the task much harder, i.e., if we first apply color jittering and then background invariance, it will be challenging to correctly identify the plants, since the color distortion inhibits a correct computation of the vegetation mask. Focusing on the highest-performing combinations we see that swapping them leads to lower values, confirming that the order in which they are applied is a key aspect when designing the augmentation policy. On the diagonal, where we apply only one augmentation, the mIoU values are all in the mid-lower range, the highest values being the strongest augmentations. This pattern is also visible in the other combinations; strong augmentations such as random erasing and mixing lead to better performance, not always at the price of longer training time.

As explained in Sec. 4.1.2.4, we changed the parameters for the color jittering augmentation. To evaluate whether this choice makes a difference, we pre-train our encoder using color jittering as the only data augmentation and fine-tune the weights at different pre-training epochs for the semantic segmentation task. We demonstrate in Tab. 4.2 that a longer pre-training period with the standard color augmentation hurts the performance. One reason could be that the augmentation is so strong that the network learns to ignore the color information, making the semantic segmentation task harder. Our augmentation consistently improves the performance the more we pre-train the encoder, outperforming the architecture pre-trained with the color jittering from the literature.

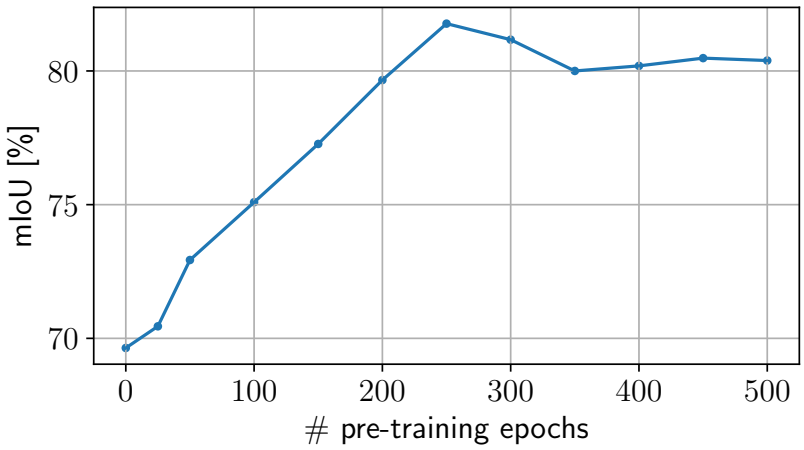


Figure 4.8: The mIoU after fine-tuning for 100 epochs using different pre-training epochs. There is no improvement after 250 epochs. The starting value is obtained with a randomly initialized network.

4.2.5 Influence of Pre-training Length

We pre-train up to 500 epochs, evaluating intermediate checkpoints on the semantic segmentation task. We want to determine how many epochs are needed to achieve satisfying results and to find out how much we can improve by continuing training. This enables us to find a compromise between performance and computational resources. In Fig. 4.8 we show that the mIoU increases until 250 epochs, where we see a diminishing effect. Therefore, if not otherwise specified, we pre-train for 250 epochs in our experiments.

4.2.6 Ablation on Augmentations' Probabilities

We apply each augmentation with a probability based on the results of the previous experiments to increase variance. We evaluate the effectiveness of our policy comparing it against a pre-training in which all augmentations are always applied. The results in Tab. 4.3 show that the probabilities we assign to each augmentation result in higher performance on all metric evaluations. We propose to assign a probability of 1.0 to color jittering and random erasing, 0.9 to gaussian blur and mixing, and 0.8 to background invariance and affine transform.

4.3 Discussion

Pre-training methods often rely on large-scale general-purpose datasets and augmentations. While such models often provide a good initialization for perception tasks, they are not tailored to capture the visual differences of narrow domains

Table 4.3: Comparison of mIoU, mean average precision (mAP), and mean average recall (mAR) after fine-tuning for 100 epochs (semantic segmentation) with our augmentation probabilities vs. applying always all the augmentations.

Approach	**mIoU** [%]	**mAP** [%]	**mAR** [%]
all augmentations	78.09	88.61	87.68
our policy	**81.77**	**91.07**	**89.99**

such as the agricultural one. This domain gap compromises the final performance of the models, especially when the size of the labeled fine-tuning dataset is small. The use of general augmentations may also distort relevant features, i.e., color or geometric cues, which could otherwise help the final perception task.

We present a domain-specific task-agnostic self-supervised pre-training strategy based on Barlow Twins, focusing on the development of augmentations that preserve relevant information for agricultural perception tasks. We use unlabeled data collected from different robotic platforms across various field conditions, which provides broad domain coverage even if without manual annotation. Our augmentation policy combines general augmentations from the literature with novel domain-specific transformations that reflect real-world perturbations. Background invariance, mixing, and controlled color jittering help simulate shadows, leaf motion, and lighting variation. Our objective is to reduce the labeling effort in agriculture by learning robust and transferable representations that improve performance even when fine-tuned on small-scale labeled datasets.

Our experimental evaluation shows improved performance across all label regimes. Our pre-training consistently outperforms random initialization, achieving the same performance using only 10% of the data, and the model pre-trained on ImageNet, even if the dataset we pre-train on is 70 times smaller. We evaluated our approach by fine-tuning the model for semantic and leaf instance segmentation, demonstrating that the learned representations are transferable over multiple perception tasks at different granularity levels. In Fig. 4.7, we show the importance of incorporating prior knowledge in the augmentation policy, even if strong general-purpose augmentations achieve comparable performance. Interestingly, the comparison between the standard color transformation and our revised version reveals that a stronger, albeit unrealistic, augmentation yields better performance for shorter pre-trainings. However, the more we pre-train with our augmentation, the more the performance improves, while the standard augmentation degrades performance. This confirms the need to align the pre-training data and augmentation policy with the downstream domain and tasks.

Although we were able to reduce the need for labeled data in both tasks, the experiments highlight that the gains are smaller for the leaf instance segmenta-

tion task, which requires more fine-grained spatial reasoning. This is expected since we are only pre-training the encoder, which extracts a very compressed representation of the images. To pre-train the entire architecture and capture sufficient spatial detail, we would need to define a new loss over the up-sampled image computed by the decoder. This is not trivial when pre-training without a specific task in mind, since the effectiveness of the pre-training depends on the alignment between the new loss and the final task.

In principle, such a strategy could lead to improvements, especially for semantic segmentation, where the task is relatively simpler and already benefits from the encoder pre-training. For example, one could pre-train using vegetation masks as pseudo-labels and later fine-tune with more accurate manual annotations. However, this approach would offer limited benefit for the instance segmentation task. We would need to modify the architecture to predict a different type of outputs, i.e., pixel-level offsets or instance centers instead of binary soil-vegetation labels. As a result, we would still be limited to pre-training only part of the architecture rather than the full model.

4.4 Conclusion

In this chapter, we presented an approach to exploit a vast quantity of unlabeled images from the agricultural domain to learn useful representations in a self-supervised fashion. Our experiments rely on domain-specific data and domain-specific augmentations during the pre-training. This allows us to successfully use our pre-training for different downstream tasks obtaining good performance using fewer labeled images. We implemented and evaluated our pre-training on two tasks, semantic and leaf instance segmentation in the agriculture domain. We compared our results with those obtained without pre-training and pre-training on ImageNet. The experiments suggest that pre-training on a domain-specific dataset and exploiting domain knowledge to define the augmentation policy can reduce the number of labeled data required to achieve the same performance as without pre-training. However, the performance gains on the leaf instance segmentation task were lower than those of the semantic segmentation. This is because of the more challenging task and the task-agnostic nature of our pre-training, which does not encode the fine-grained spatial details needed for separating the leaves. These observations motivate the next chapter, where we shift our focus on semantic segmentation, a foundational task for agricultural scene understanding. We propose an automatic labeling pipeline that can better exploit the knowledge from the agricultural setting given the fixed task and paves the way for more scalable and effective perception systems.

Chapter 5

Exploiting Spatial Arrangement for Semantic Segmentation

Semantic segmentation of images is the task of assigning a class label to every pixel of the image. In agriculture, it can be used to distinguish which parts of the image correspond to soil, crop, or weed. This pixel-level understanding is critical for automating tasks such as automated weeding [12, 192], controlled usage of pesticide [156] to apply them in areas with high weed density and not uniformly over the whole field, and harvesting [166]. Semantic segmentation is also the first step of more complex tasks such as yield estimation [54], growth monitoring [144], and disease detection [169].

In the previous chapter, we demonstrated how to boost the performance of semantic segmentation by exploiting our domain-specific self-supervised pre-training. However, the network still needs access to some manually labeled data to learn the semantic segmentation task. In this chapter, we examine the problem of automated semantic crop-weed segmentation in RGB images under various field deployment conditions, e.g., different growth stages, crop species, or lighting conditions, without human-labeled training data. This is crucial for ensuring robust crop-weed segmentation in unseen fields, enabling robots to perform weeding and harvesting. Our approach automatically labels RGB images based on the robot's pose and its current semantic map of the field, which is computed online using the automatic labels of previous RGB images. In this way, semantic labels are generated using the robot's spatial information and the spatial arrangement of the field, i.e., its crop row structure.

Previous methods for unsupervised semantic segmentation in agriculture proposed by Lottes et al. [133] and Winterhalter et al. [226] are heuristic-based and rely on assumptions about field arrangements that do not necessarily generalize well, e.g., absence of weeds in the crop row [133], constant distance between plants' rows [134, 226], or non-overlapping vegetation components [133]. In con-

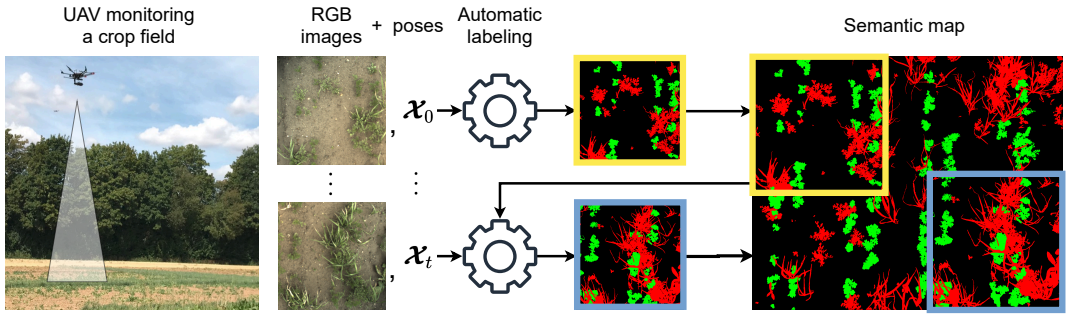


Figure 5.1: The overview of our pipeline to generate semantic labels for images of crop fields. We use a robotic platform equipped with an RGB camera to collect images of the field with their corresponding poses. We process each image and generate its semantic segmentation, which we then fuse into the semantic map. At each time step, we use the semantic map to generate the image's semantic label and then, we update the map accordingly.

trast, fully supervised deep learning-based approaches do not rely on heuristics, but on in-domain human-labeled training data. The performance of such methods is satisfactory when deployed in conditions they were exposed to during training. However, their performance usually rapidly deteriorates in novel deployment conditions [180], e.g., different crop species, weed pressure, lighting conditions, or growth stage, requiring new human-labeled training data. This additional labeling is costly and limits the fully supervised approaches when there is not enough time, money, or data to re-train the approach on new field conditions.

The main contribution of this chapter is a novel heuristic approach for unsupervised soil-weed-crop segmentation in managed fields with crop row structure, which we exploit to address the limitations of previous works. Our method automatically generates labels used to train deep semantic segmentation networks. The overview of our pipeline is shown in Fig. 5.1. Our pipeline takes the current RGB image and the pose of the robotic platform as input to compute a semantic map of the field. As a key novelty, we use the semantic map to enforce the spatial consistency of labels. To this end, we leverage the information about the crop rows in the map to enhance crop segmentation across different growth stages. We additionally do not assign labels to vegetation components that are near the crop rows but have not been classified as crops, i.e., they do not lie directly on the crop row. This reduces labeling errors, enabling the generation of high-quality labels and thus improving model predictions after training on our generated labels. We use the labels generated by our heuristic approach to train an uncertainty-aware evidential semantic segmentation network [194]. At inference, as a post-processing step, we exploit the predicted uncertainties to refine the final semantic predictions.

Our experimental evaluation demonstrates that our approach produces more accurate semantic labels than previous unsupervised label generation approaches on multiple crop species, growth stages, and lighting conditions. Our approach outperforms previous approaches thanks to the combination of our spatially consistent generated labels and an uncertainty-aware semantic neural network. We improve the performance of fully supervised models on previously unseen crops, growth stages, or soil conditions by fine-tuning them on labels automatically generated with our approach for the new images.

5.1 Uncertainty-Aware Networks

Classical neural networks are known to provide overconfident point estimate predictions [1], i.e., they provide a single "best guess" for each pixel and overestimate the confidence in their prediction even when the prediction is not correct. Several works, including the one by Lakshminarayanan et al. [117], use ensembles of multiple independently initialized and trained neural networks to quantify predictive uncertainty. Although ensembles improve prediction performance and model calibration, they induce high computational costs during training. Gal et al. [65] proposed Monte Carlo dropout to approximate predictive uncertainty with a single network trained with dropout. At inference, they perform multiple forward passes with independently sampled dropout masks to compute predictive uncertainty. Although Monte Carlo dropout is more computationally efficient during training, it produces more overconfident predictions than ensembles [15]. More recently, Sensoy et al. [194] proposed evidential deep learning for image classification to predict uncertainty using a single forward pass. As evidential deep learning performs on par with ensembles while drastically reducing online compute requirements, we use it as part of our semantic segmentation network. We leverage the predicted uncertainties in the post-processing of our predicted labels to correct our prediction for the weed class, which is the most under-represented class and, thus, the most uncertain for the model.

5.2 Our Approach for Automatic Soil-Weed-Crop Segmentation

We propose a heuristic-based approach to automatically segment RGB images of agricultural fields into three semantic classes: soil, crop, and weed. Sec. 5.2.1 describes the fusion of each generated semantic label into an online-built global semantic field map based on the robot's pose. Then, Sec. 5.2.2 presents our automatic labeling pipeline, which enforces spatial label consistency while reducing

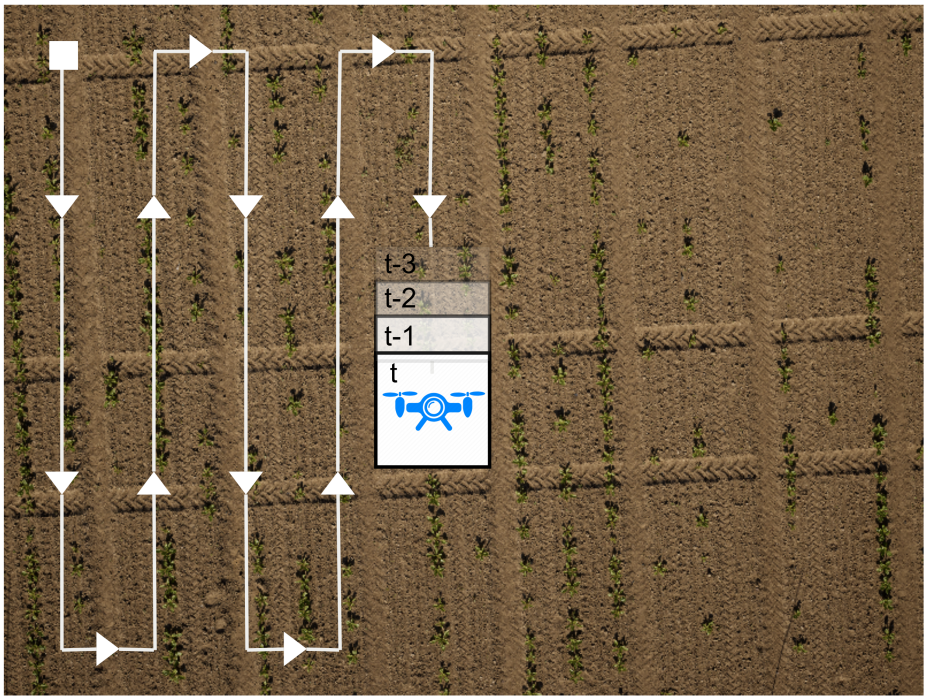


Figure 5.2: Example of a typical UAV mission. The coverage path along which we fuse semantic image labels is depicted in white, the square is the initial pose, and the arrows indicate the direction of movement. The images can overlap, but it is not required. This path ensures efficient coverage of the crop field and is commonly adopted in aerial data collection missions.

the possibility of labeling errors in the map. Sec. 5.2.3 describes the training procedure of our uncertainty-aware semantic segmentation network [194] on labels extracted from the global semantic field map. Finally, Sec. 5.2.4 outlines our uncertainty-based post-processing for refining uncertain vegetation predictions.

5.2.1 Semantic Field Mapping

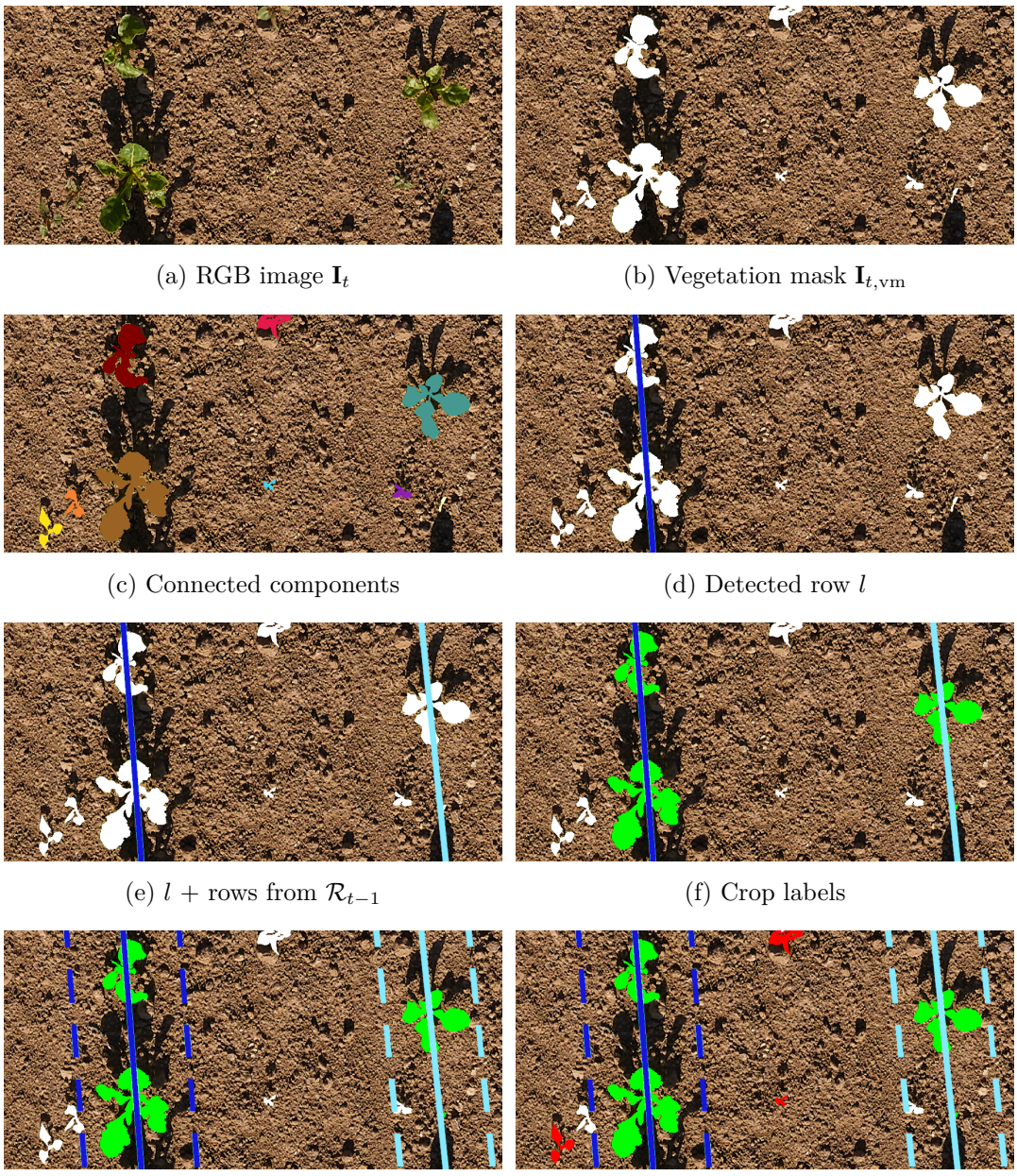
We perform semantic mapping to enforce spatial consistency across automatically generated semantic labels. Furthermore, the semantic map allows us to extract image-label pairs from the map with different rotations, positions, and scales. We assume that our robotic system is equipped with a downwards-facing RGB camera. At each time step t, the robotic platform collects an image $\mathbf{I}_t \in \mathbb{R}^{H \times W \times 3}$, where H and W are the height and width of the image, respectively. Let $\boldsymbol{\mathcal{X}}_t = (x_t, y_t, z_t, \phi_t)^\top$ be the robot pose, where we consider the 3D position (x_t, y_t, z_t) and the yaw angle $\phi_t \in (-\pi, \pi]$ with respect to the origin of the mapping mission. Any path is defined by a sequence of poses that we use to fuse our predicted labels into the global semantic field map $S_t : G \rightarrow \mathbb{N}^{K \times \hat{H} \times \hat{W}}$, where G is a grid discretizing the environment into $\hat{H} \times \hat{W}$ cells with K possible seman-

tic classes. Each image $\mathbf{I}_t$ along the path is segmented by our approach based on the previous map S_{t-1} and then fused into the semantic map to compute S_t accumulating predictions. We use majority voting to assign the most likely class when images overlap, leading to multiple predictions for the same map pixel. In practice, we follow a common lawnmower-like coverage path to efficiently cover agricultural fields [94], as shown in Fig. 5.2.

5.2.2 Automatic Labeling

At each time step t, our automatic labeling approach takes as input the image $\mathbf{I}_t$, the current pose $\boldsymbol{\mathcal{X}}_t$, and the semantic field map S_{t-1} to produce a semantic label for image $\mathbf{I}_t$. We use the map S_{t-1} to estimate potential weeds and crops in $\mathbf{I}_t$ enforcing spatial consistency to reduce labeling errors. This is possible because the semantic field map contains information about the set of crop rows detected in previous images, called $\mathcal{R}_{t-1}$. Our automatic labeling procedure is exemplarily visualized in Fig. 5.3 and consists of the following steps: first, we extract the vegetation mask and apply the Hough transform to detect the main crop row in the current image $\mathbf{I}_t$. Second, we propagate all previously detected crop rows $\mathcal{R}_{t-1}$ to the current pose to segment multiple crop rows. All vegetation components intersecting a line are labeled as crops. Third, we label the vegetation components with a minimal distance to all rows as weeds.

Hough transform. We compute the binary vegetation mask $\mathbf{I}_{t,\mathrm{vm}} \in \{0,1\}^{H \times W}$ visualized in Fig. 5.3 (b) via graph-based image segmentation [62], where a pixel is 1 if it contains vegetation, i.e. crop or weed, and 0 otherwise. We apply the Hough transform introduced by Hough et al. [90] to the vegetation mask $\mathbf{I}_{t,\mathrm{vm}}$ to detect crop rows in image $\mathbf{I}_t$. This gives us a set of supporting lines in $\mathbf{I}_t$. Each line l is parameterized by the distance $r_{t,l}$ from the image origin to the line, and the angle $\theta_{t,l}$ between the image's x-axis and the line connecting the origin to the closest point on the line. The origin is the lower-left pixel of $\mathbf{I}_t$. The best-fitting line is the one that maximizes the overlap with the vegetation mask $\mathbf{I}_{t,\mathrm{vm}}$. We show in Fig. 5.4 an example of a fitted crop row line (white). We discretize the Hough line radius search space using a pixel resolution of $l_w = 5\,\mathrm{px}$ to fit lines robustly even in the presence of noisy vegetation masks. We define the minimum number of overlapping pixels $\tau_{px} = H$ to fit the line along the whole image height if the crop rows are aligned with the y-direction of the camera, and $\tau_{px} = W$ otherwise. We keep only the best-fitting line of parameters (r_t, θ_t) returned from the Hough transform and add it to the set of the crop rows detected in the map $\mathcal{R}_t = \mathcal{R}_{t-1} \cup (r_t, \theta_t)$ to use them in the following step. Based on the best-fitting line parameters (r_t, θ_t), we create a binary mask $\mathbf{I}_{t,\mathrm{line}}$, which is 1 for all pixels on the line and 0 otherwise. We transform the line parameters for this time step t



(a) RGB image $\mathbf{I}_t$ (b) Vegetation mask $\mathbf{I}_{t,\text{vm}}$

(c) Connected components (d) Detected row l

(e) l + rows from $\mathcal{R}_{t-1}$ (f) Crop labels

(g) Uncertain areas (h) Final label

Figure 5.3: Steps of our automatic labeling approach for the $\mathbf{I}_t$ image in (a). In (b), the extracted vegetation mask $\mathbf{I}_{t,\text{vm}}$ (white). In (c), we show the connected components, and in (d) the most prominent crop row detected via the Hough transform (dark blue). In (e), we see the result after propagating the set of previously detected crop rows $\mathcal{R}_{t-1}$ (light blue) into the current image. (f) shows all components labeled as crops (green). In (g), we see the area of uncertainty around each crop row. Finally, in (h), we show the final segmentation, where components outside the uncertain area are labeled as weeds (red). All the components with at least one pixel inside the uncertain areas are assigned to the "unknown" class (white).

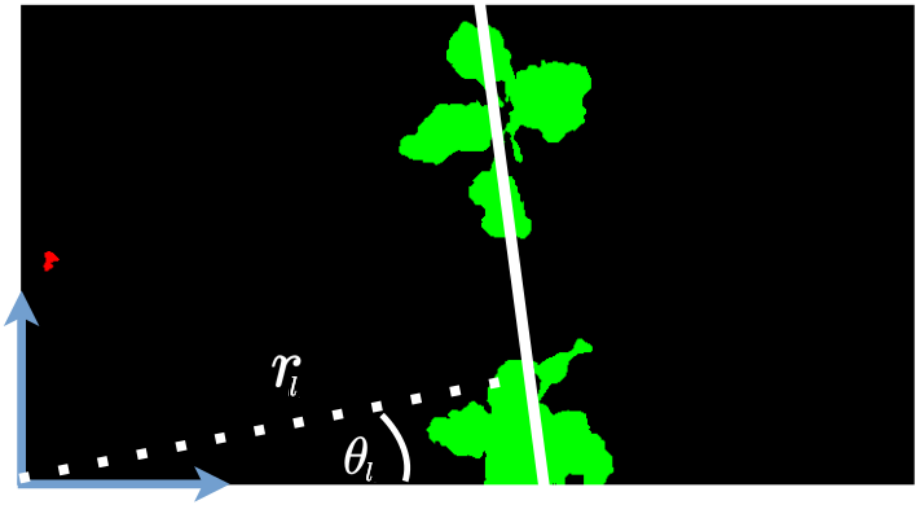


Figure 5.4: Given the vegetation mask, we visualize the line detected by the Hough transform (in white). Considering the origin as the bottom left corner, we show the parameters r_l and θ_l defining the detected line l. The vegetation components intersecting the line are thus labeled as crop (green). We label the vegetation component on the far left as a weed (red), since it is far from the detected line.

into the coordinate system of the mapping mission's origin $\boldsymbol{\mathcal{X}}_0$, as

$$r_{t,l} = \left\| \left(\mathbf{T}_t^0\right)^{-1} \begin{bmatrix} r_{t-1,l}\cos(\theta_{t-1,l}) \\ r_{t-1,l}\sin(\theta_{t-1,l}) \\ 0 \end{bmatrix} \right\|_2, \tag{5.1}$$

$$\theta_{t,l} = \theta_{t-1,l} - \phi_t, \tag{5.2}$$

where $r_{t-1,i}\cos(\theta_{t-1,l})$ and $r_{t-1,l}\sin(\theta_{t-1,l})$ represent the (i,j) coordinates of the closest pixel to the origin of $\mathbf{I}_t$ for line l, assuming flat terrain. We save the line mask to facilitate the computation of the following steps. The mask obtained from our example image is shown overlayed to the vegetation mask in Fig. 5.3 (d).

Propagating predictions. We use our semantic map S_{t-1} to retrieve the predicted lines $\mathcal{R}_{t-1}$ and propagate them into our current image $\mathbf{I}_t$. This enables the prediction of multiple crop rows consistent with the rows detected in previously explored areas of the crop field. When $t = 0$, the semantic map and $\mathcal{R}_{t-1}$ are both empty and we can skip this step. At each time step $t \geq 1$, we first compute the position of the newly acquired image in the coordinate system of the initial pose $\boldsymbol{\mathcal{X}}_0$, given by the transformation matrix $\mathbf{T}_t^0 \in \mathbb{R}^{3\times3}$. Then, we check which lines in $\mathcal{R}_{t-1}$ intersect $\mathbf{I}_t$ and should be propagated into its semantic prediction using Eq. (5.1) and Eq. (5.2).

We include these lines in $\mathbf{I}_{t,\text{line}}$, i.e. we set the pixels covered by these lines to 1. To reduce the computation time, we reject lines that are too close to those already present in the mask $\mathbf{I}_{t,\text{line}}$. In particular, we reject line l if its distance

to any other line in $\mathcal{R}_t$ is smaller than $2l_w$. In Fig. 5.3 (e), we showcase line propagation from a previous image, enabling us to detect a second crop row on the image's right side.

As we propagate our line predictions from previously recorded images into the current image, we use an eroded version the vegetation mask $\mathbf{I}_{t,\text{vm}}$ to extract single vegetation components. We use a square kernel of size 3 for the erosion to remove noise from $\mathbf{I}_{t,\text{vm}}$ and reduce the mislabeling of weeds touching the crops in the crop row. We then apply standard connected component analysis [22] on the eroded mask to isolate distinct vegetation components, which we show in Fig. 5.3 (c). Then, all components intersecting lines in $\mathbf{I}_{t,\text{line}}$ are assigned to the crop class, yielding a new binary mask $\mathbf{M}_t \in \{0,1\}^{H\times W}$ where a pixel is 1 if it is labeled as crop, and 0 otherwise. We show the result in Fig. 5.3 (f), where crops are colored in green, and the rest of the vegetation components are still in white. Next, we describe which remaining vegetation components are assigned to the weed class.

Weed labeling. Naively classifying any vegetation component in $\mathbf{I}_{t,\text{vm}}$ not yet labeled as crops in $\mathbf{M}_t$ usually results in poor weed label quality. These remaining vegetation components may still be crops, not labeled because of the failure of the row detection due to low sensor resolution, wrong odometry or pose information, or bad lighting conditions [132]. To avoid labeling these potential crops as weeds, we do not label the vegetation components which are likely to introduce labeling errors, and ignore them during network training. To this end, we compute the distance from each of the N crop pixels of $\mathbf{M}_t$ with value 1 to their respective closest line as follows

$$d(i,j) = \arg\min_{(r_{t,l},\theta_{t,l})\in\mathcal{R}_t} \left| i\cos(\theta_{t,l}) + j\sin(\theta_{t,l}) - r_{t,l} \right|. \tag{5.3}$$

We aim to estimate crop sizes along the detected rows using these distances. Hence, we use an indicator function $\mathbb{1}(i,j)$ that returns 1 if the pixel (x,y) is 1 in $\mathbf{M}_t$ and zero otherwise to extract the mean $\mu_d = \frac{\sum_{(i,j)} \mathbb{1}(i,j)\, d(i,j)}{N}$ and standard deviation $\sigma_d = \sqrt{\frac{\sum_{(i,j)} \mathbb{1}(i,j)\, (d(i,j)-\mu_d)^2}{N}}$. We define the minimum distance $d_{\min}$ required for any unlabeled vegetation instance to be labeled as a weed as

$$d_{\min} = \mu_d + \delta\, \sigma_d, \tag{5.4}$$

where $\delta = 3$ in our setting, such that only vegetation instances with a large distance from all rows are considered weeds. We show the uncertain areas around the crop rows in Fig. 5.3 (g). All vegetation components that were not labeled as crops and whose distance to the lines is smaller than $d_{\min}$ are left unlabeled, i.e., they are assigned to an "unknown" class. Note that large values of δ reduce the number of components labeled as weeds, while small values of δ are prone to weed labeling errors. The key idea behind this step is that we use μ_d and σ_d

to represent the area around the detected crop row where we assume there may be other crops that were not touched by the line and that we leave unlabeled. Outside of this area, we are fairly confident that the vegetation component is a weed as it is far from the detected crop row with plants of estimated size μ_d. The resulting label for the example image is shown in Fig. 5.3 (h), where components close to the crop row on the right are not labeled while the component on the upper-left corner is labeled as a weed.

5.2.3 Learning with Uncertainty

Once we finish our mapping mission as described in Sec. 5.2.2, we can extract any number of image-label pairs with any size, rotation, and aspect ratio. We use the extracted labels to train a semantic segmentation network, which will predict a semantic class also for the pixels currently assigned to the "unknown" class. We follow the evidential deep learning framework by Sensoy et al. [194] to predict at the same time both semantic segmentation and the network's uncertainty. Estimating the prediction uncertainty enables to account for the "unknown" class by refining the network's semantic predictions in a post-processing step described in Sec. 5.2.4.

The key idea behind evidential deep learning is to predict a Dirichlet distribution over all possible class probabilities instead of a single point estimate as in deterministic deep neural networks. In this way, the evidential network minimizes the prediction error while maximizing the prediction uncertainty for ambiguous image parts. We use evidential deep learning instead of Bayesian deep learning approaches [15, 65] as it is empirically shown to produce similarly or better-calibrated prediction uncertainties [194] while being computationally more efficient during training than ensemble methods and during inference than Monte Carlo dropout.

We train the network to minimize for every pixel (i, j) of image $\mathbf{I}$ the Bayes risk cross-entropy

$$\mathcal{L}_{CE,(i,j)} = \sum_{k=1}^{K-1} \mathbf{y}_{(i,j),k} \left(\psi(Q_{(i,j)}) - \psi(\boldsymbol{\alpha}_{(i,j),k}) \right), \tag{5.5}$$

where ψ is the digamma function, $\mathbf{y}_{(i,j),k} = 1$ if the pixel (i, j) of $\mathbf{I}$ belongs to ground truth class k, $Q_{(i,j)} = \sum_{k=1}^{K} \boldsymbol{\alpha}_{(i,j),k}$, and $\boldsymbol{\alpha}_{(i,j),k}$ is the evidence predicted by the network in support of class k. We do not compute this loss for the pixels assigned to the "unknown" class, so we sum only over the remaining $K - 1$ classes, i.e., soil, crop, and weed. We additionally minimize the Kullback-Leibler (KL) divergence between the uniform $D(\mathbf{1}_{K-1})$ and predicted $D(\tilde{\boldsymbol{\alpha}}_{(i,j)})$ Dirichlet distribution for all non-ground-truth classes [194],

$$\mathcal{L}_{(i,j)} = \mathcal{L}_{CE,(i,j)} + \lambda_{\text{epoch}} \text{KL}[D(\tilde{\boldsymbol{\alpha}}_{(i,j)}) || D(\mathbf{1}_{K-1})], \tag{5.6}$$

$$\tilde{\boldsymbol{\alpha}}_{(i,j),k} = \mathbf{y}_{(i,j),k} + (1 - \mathbf{y}_{(i,j),k})\boldsymbol{\alpha}_{(i,j),k}, \tag{5.7}$$

for all $K-1$ classes, and $\lambda_{\text{epoch}} = \min(1.0, \frac{\text{epoch}}{T})$ with epoch being the current training epoch and T the maximum annealing epoch. We minimize the overall training loss

$$\mathcal{L} = \frac{1}{HW} \sum_{i=1}^{H} \sum_{j=1}^{W} \mathcal{L}_{(i,j)}, \tag{5.8}$$

which is the average over all image pixels, iterating over all training images. At inference time, the network predicts the semantic class and an uncertainty for each pixel, that we use for our label refinement.

5.2.4 Uncertainty-Based Label Refinement

We use the network's predicted Dirichlet distribution $D(\boldsymbol{\alpha}_{(i,j)})$ over all $K-1$ classes to quantify the prediction uncertainty for post-processing and refining the predicted semantic labels. For a pixel (i,j) of image $\mathbf{I}$, the network's prediction uncertainty [194] is given by

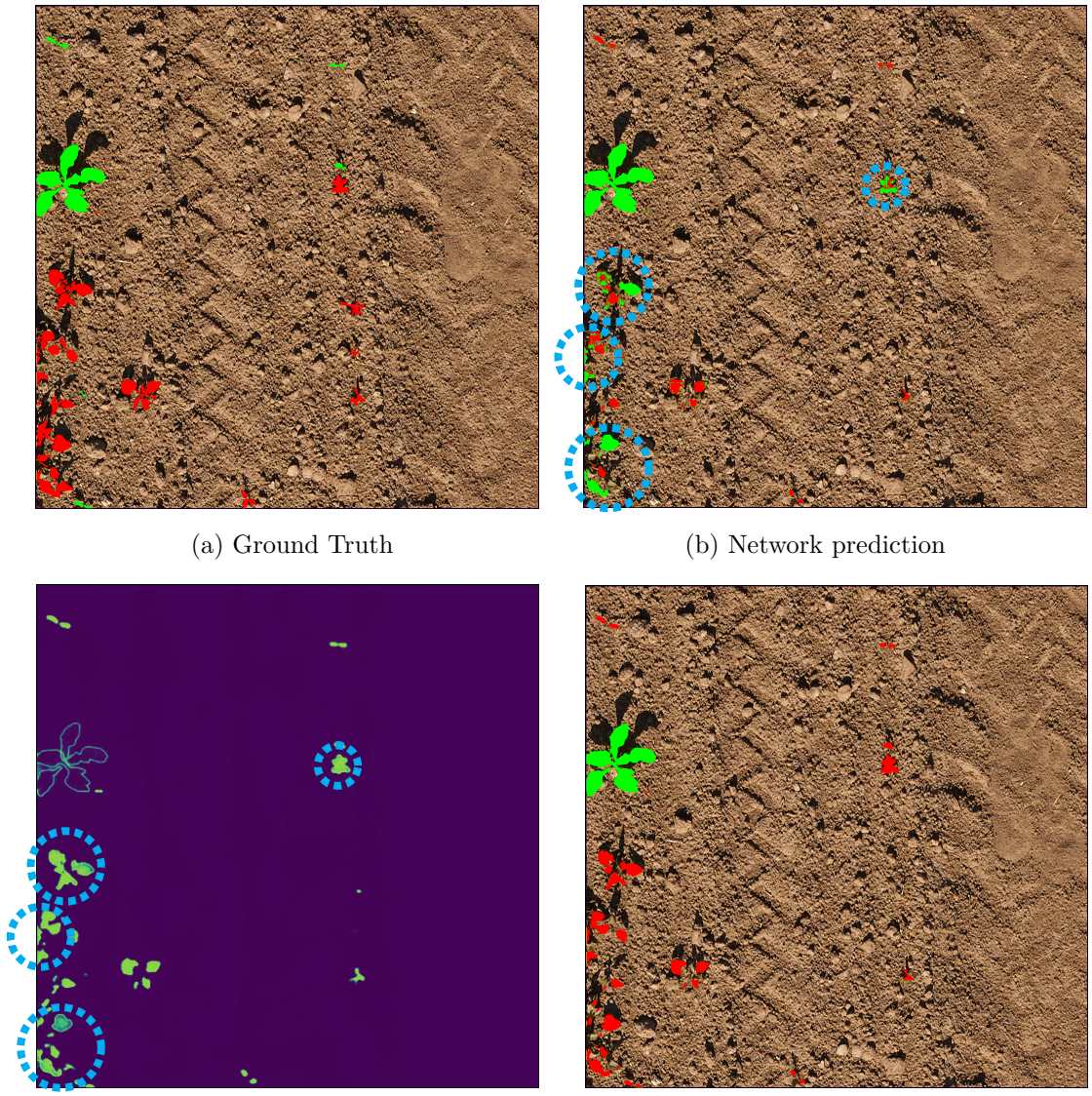
$$u_{t,(i,j)} = \frac{K-1}{\sum_{k=1}^{K-1} \boldsymbol{\alpha}_{(i,j),k}}, \tag{5.9}$$

where $K-1$ is the number of classes excluding the "unknown" class. In our crop-weed segmentation case, the most under-represented class is weed. Thus, the network will likely be more uncertain about areas representing weeds than the other classes. We define an adaptive threshold τ to select the most uncertain pixels (i,j) in any image $\mathbf{I}$ as

$$\tau = \frac{\max(\mathbf{U}_t) - \min(\mathbf{U}_t)}{2} + \min(\mathbf{U}_t). \tag{5.10}$$

We compute a binary mask $\mathbf{U}_t \in \{0,1\}^{H \times W}$ where a pixel (i,j) is 1, if $\mathbf{U}_t(i,j) > \tau$, and 0 otherwise. We compute the connected components [22] of the pixels predicted as crops from the network, aiming to use the ratio between the size of the object and its number of uncertain pixels to refine the component's label. Most of the components have high uncertainty at their instance boundaries. However, we are interested only in the components with large amounts of uncertain interior pixels. We iterate over all crop components $cc \in \{1, \ldots, C\}$ in our prediction and compute a binary mask $\mathbf{C}_{cc} \in \{0,1\}^{H \times W}$ for each component, which is 1 for all pixels belonging to the component. We also compute the components' bounding box $\mathbf{b}_{cc} = (b_{cc}^x, b_{cc}^y, b_{cc}^{\text{height}}, b_{cc}^{\text{width}})$, where b_{cc}^x and b_{cc}^y are the coordinates of the upper left corner of the bounding box, while b_{cc}^{height} and b_{cc}^{width} are the height and width of the bounding box. We define an adaptive threshold

$$\tau_{cc} = \frac{1}{4} \min\left(\frac{b_{cc}^{\text{width}}}{b_{cc}^{\text{height}}}, \frac{b_{cc}^{\text{height}}}{b_{cc}^{\text{width}}} \right). \tag{5.11}$$



(a) Ground Truth

(b) Network prediction

(c) Uncertainty

(d) Final prediction

Figure 5.5: We illustrate how we polish the prediction of the network for an exemplary image. In (a), we show the ground truth labels over the RGB input image, where weeds are colored in red, and crops are colored in green. In (b), we show the prediction of the network and highlight the errors with white dotted circles. In (c), we can see the uncertainty of the network, where highly uncertain areas are colored in green-yellow. As expected, the network is uncertain about the boundaries of the plants, which are always hard to classify correctly, and about the weeds. Even the weeds already classified as crops by the network have high uncertainty. In (d), we show the final prediction after our post-processing, where we label as weeds the highly uncertain vegetation components. This corrects many of the network's errors.

This threshold helps us avoid detecting as weeds small spikes of uncertainty that could arise because of shadows, reflections, or insects. In this way, we only act upon vegetation components where there is a large uncertain area. If the network is uncertain about the prediction of crop component cc, it holds that

$$\frac{\sum_{(i,j)} \mathbf{U}_{(i,j)} \mathbf{C}_{cc,(i,j)}}{b_{cc}^{\text{width}}\ b_{cc}^{\text{height}}} > \tau_{cc}. \tag{5.12}$$

If crop component cc fulfills Eq. 5.12, we assign the component's uncertain pixels (i, j) with $\mathbf{U}_{(i,j)} = 1$ to the weed class. We do not re-assign the whole vegetation component because our network does not provide instances. Hence, there may be components that contain both weeds and crops, i.e., when crops and weeds share a boundary and are not separated by the soil. We show in Fig. 5.5 the result of our post-processing operation for an example image, highlighting the correspondence between the network's wrong predictions, the estimated uncertainty, and the post-processed semantic prediction.

5.3 Experimental Evaluation

The main focus of this chapter is an automatic labeling pipeline for semantic soil-weed-crop segmentation of RGB images. In the following experiments, we show how we generate more accurate semantic labels than previous unsupervised label generation approaches on multiple crop species, growth stages, and lighting conditions. Our approach outperforms previous approaches thanks to the combination of our spatially consistent generated labels and an uncertainty-aware semantic neural network. We improve the performance of fully supervised models on previously unseen crops, growth stages, or soil conditions after fine-tuning them on our automatically generated labels.

5.3.1 Experimental Setup

Datasets. We employ four datasets, three of which are publicly available: PhenoBench [224], as well as the Carrots and Onions datasets from Lincoln University [20]. The last one is the SugarBeets dataset [225] introduced in Chapter 3. The Carrots dataset was recorded in Lincolnshire, UK, in June. The field is under substantial weed pressure and contains weeds with a similar appearance to the crop. Furthermore, several regions contain crops and weeds in close proximity to each other. The Onions dataset was also recorded in Lincolnshire, UK, but in April. The weed pressure is lower compared to the Carrots dataset. The PhenoBench dataset was recorded in Meckenheim, Germany, on different dates between May and June to capture different growth stages. The field contains two varieties of sugar beets and six different weed varieties. The weed pressure varies

Table 5.1: Details for all the used datasets: name, reference paper, camera sensor, image resolution and GSD.

Dataset	Camera	Image Resolution [px]	GSD $\left[\frac{\text{mm}}{\text{px}}\right]$
PhenoBench [224]	PhaseOne iXM-100 with a 80 mm RSM prime lens on a gimbal (UAV)	$11\,664 \times 8\,750$	1
Carrots [20]	Teledyne DALSA Genie Nano deployed on a manually pulled cart (UGV)	$2\,428 \times 1\,985$	0.4
Onions [20]	Teledyne DALSA Genie Nano deployed on a manually pulled cart (UGV)	$2\,149 \times 1\,986$	0.4
SugarBeets [225]	PhaseOne iXM-100 (UAV)	$4\,320 \times 4\,100$	1.5

Table 5.2: List of the hyperparameters of our method, where they are used, and their chosen values.

Hyperparameter	Method	Value
number of pixels for detection (τ_{px})	Hough line detection	H (image height)
width of the line to fit (l_w)	Hough line detection	5 px
confidence interval for crop rows (δ)	Weed labeling	3
number of annealing epochs (T)	Evidential Deep Learning	25

as the dataset contains images from fields that were fully, partially, or not treated with herbicides at all. Finally, the SugarBeets dataset was also recorded in Meckenheim, Germany, over five different weekly sessions. The field is arranged with small spacing between plants and high weed pressure, inducing challenging conditions. We refer to Tab. 5.1 for information about the camera, image resolution, and ground sampling distance of the datasets.

Metrics. As in Chapter 3 and in Chapter 4, we use the IoU as a metric for semantic segmentation. For the automatic labeling pipeline, we also report the boundary IoU [40] to have a better understanding of the approaches' limitations. While the traditional IoU evaluates the overlap between predicted and ground-truth regions, the boundary IoU is computed only on the objects' contours.

Training details and hyperparameters. We use ERFNet [184] as our network and train it using the Adam [107] optimizer, a learning rate of 0.01, and a batch size of 32. We set $T = 25$ in Eq. 5.6 to linearly increase λ_{epoch} over the first 25 epochs. We report all the hyperparameters of our method with their values in Tab. 5.2. To evaluate the quality of the labels generated by the approaches, we use them to generate labels for the validation sets of PhenoBench and SugarBeets, as well as for the whole Carrots and Onions datasets. For PhenoBench and

SugarBeets, we automatically generate labels for the images in their training sets to train our network and evaluate the results on the manually annotated validation sets. We do not split Carrots and Onions to train on them since they consist of only 20 images each. Thus, we do not use them for model training. Instead, we evaluate our label generation and the generalization capabilities of models fine-tuned on these datasets.

Baselines. We use three baselines: two are general-purpose unsupervised semantic segmentation networks not specifically developed for the agricultural domain, and one is an automatic labeling method specifically developed for the agricultural domain. The first baseline is STEGO [77], which provides an official implementation for the evaluation alongside the weights of their models. We use the model based on vision transformers [57] and trained on MS COCO [127]. STEGO predicts per-pixel features and clusters them using the mechanisms of self and cross attention [213]. Our second baseline is U2Seg [161], which builds upon STEGO and leverages instance information to overcome some of the limitations of the previous work; they also open-source their code and provide their models. U2Seg proposes an unsupervised universal segmentation approach that directly predicts clusters from which it is possible to recover both the semantic class and the instance ID. We use the model predicting 800 clusters and trained on ImageNet [52] and MS COCO. Lottes et al. [132] propose a domain-specific method for generating per-pixel crop and weed labels. They use a vegetation mask to detect the main crop row and then label all other vegetation components as weeds. We use their official implementation, removing the NIR image channels. We evaluate their automatically generated labels and the performance of ERFNet trained on their labels. We train the same ERFNet network with the same training hyperparameters on their and our generated labels to ensure a fair comparison. We report the results of ERFNet trained in a fully-supervised fashion on PhenoBench as a performance upper bound.

5.3.2 Automatic Labeling

In the first experiment, we evaluate the performance of our automatic labeling pipeline for semantic soil-weed-crop segmentation on multiple datasets. We compare against two general-purpose unsupervised semantic segmentation networks [77, 161] and the domain-specific approach by Lottes et al. [132].

We show the results on all four datasets in Tab. 5.3. The general-purpose approaches perform worse than the domain-specific methods across all datasets, except for U2Seg on the Carrots dataset. As Onions have thin leaves, they are hard to detect with common color histogram thresholding methods, such as the one by Lottes et al. [132]. Furthermore, the weeds in this dataset have a size similar to the crops, leading to bias in crop row detection and introducing a

Table 5.3: Performance of all the baselines on the PhenoBench dataset, Carrots dataset, Onions dataset, and SugarBeets dataset. The top rows are the general purpose approaches, while the bottom rows are the domain-specific ones. We report the mean IoU, plus the IoU and boundary IoU per class. In **bold** the best results per column.

Dataset	Approach	IoU [%]			mIoU [%]	Boundary IoU [%]		
		soil	crop	weed		soil	crop	weed
PhenoBench	STEGO	21.4	11.9	0.4	11.2	0.0	1.5	0.0
	U2Seg	84.6	40.0	2.4	42.3	45.8	11.7	3.4
	Lottes	**99.6**	44.1	**7.6**	50.5	0.0	0.0	0.9
	Ours	98.8	**80.7**	7.2	**62.2**	**86.3**	**79.1**	**13.2**
Carrots	STEGO	28.4	5.1	15.8	16.4	0.0	0.9	0.0
	U2Seg	80.1	**20.4**	2.3	34.3	36.2	0.0	**19.3**
	Lottes	89.1	15.9	34.0	46.3	0.0	0.0	6.8
	Ours	**90.4**	12.6	**42.7**	**48.6**	**84.4**	**23.6**	9.4
Onions	STEGO	26.5	5.1	3.0	11.5	0.0	2.4	0.0
	U2Seg	92.8	0.0	4.3	32.4	24.2	0.0	8.2
	Lottes	89.7	1.4	1.1	30.7	0.0	0.0	1.6
	Ours	**95.4**	**10.7**	**16.6**	**40.9**	**74.2**	**10.7**	**16.7**
SugarBeets	STEGO	24.9	4.7	1.3	10.3	0.0	1.9	0.0
	U2Seg	77.9	9.9	6.7	31.5	1.8	**2.8**	0.0
	Lottes	**98.0**	23.6	18.8	46.8	0.0	0.0	1.5
	Ours	97.7	**50.6**	**24.7**	**57.7**	**88.7**	0.0	**1.8**

higher risk of confusing the two classes. Our approach for automatic labeling, i.e., without the uncertainty-aware network, shows higher crop label quality than Lottes while performing on par or better in terms of weed label quality. Particularly, Lottes confuses more weeds with crops, while our approach does not assign labels to hard-to-label vegetation components, as described in Sec. 5.2.2. The Carrots dataset is the only one where U2Seg outperforms the domain-specific approaches, which suffer from the weed pressure when estimating the crop rows. Our method consistently outperforms all other baselines across all datasets covering different crop species, weed pressure, growth stages, and lighting conditions. On the Onions dataset, where most approaches fail, we improve by approx. 9% mIoU over the second-best baseline, U2Seg, due to higher IoU in both vegetation classes. We draw attention to the fact that all pixels classified as "unknown" are considered errors at this point of the evaluation.

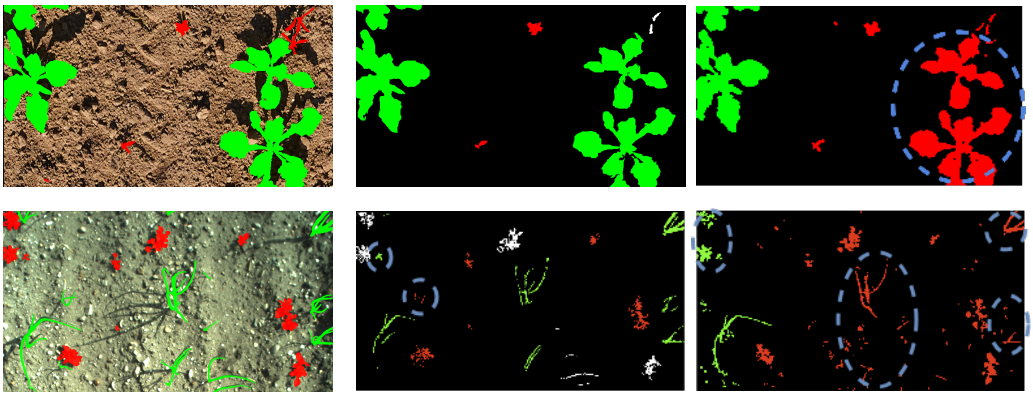


Image with Ground Truth | Ours | Lottes et al. [132]

Figure 5.6: Qualitative results of our and Lottes et al. methods on PhenoBench (top row) and Onions (bottom row). We depict soil in black, crops in green, weeds in red, and the vegetation that we leave unlabeled in white. We highlight segmentation errors in dashed blue circles.

To gain additional insights on the accuracy of the generated labels, we compute the boundary IoU. This is important for phenotyping, as accurate boundary detection helps measure the plant size and growth rate. The metric confirms the result of the standard IoU. As shown in Tab. 5.3, the approach by Lottes et al. [132] incorrectly segments boundaries on most of the datasets. This might be due to wrongly segmented vegetation masks. Aiming to include the boundary of weeds more accurately may worsen the overall performance, as the soil could be mistakenly classified as vegetation. The results suggest that our approach might suffer from the same problem on the Carrots dataset. The discrepancy between IoU and boundary IoU per class suggests that we underestimate the size of weeds, i.e., high IoU but low boundary IoU for weeds, and overestimate crop size, i.e., low IoU but high boundary IoU for crops. On the Carrots dataset, U2Seg outperforms the other methods in the weeds boundary IoU. The weed IoU suggests that U2Seg overestimates weeds, thus obtaining a boost as the total number of pixels in the IoU computation is low. On the Onions dataset, our method's IoU and boundary IoU are almost the same, irrespective of the semantic class. Since the crops and weeds are thin, the boundary area covers the whole vegetation instance. The other approaches fail to correctly assign weed and crop boundaries on the Onions dataset, which is a direct result of the low IoU on both weeds and crops. On the SugarBeets dataset, all approaches fail to predict boundaries, most likely due to unusually high weed pressure. Our method accurately segments soil boundaries, suggesting that it at least successfully differentiates between soil and vegetation. Overall, the results indicate that most approaches underestimate the size of vegetation, both crops and weeds. Instead, our automatic labeling method

Table 5.4: Performance of ERFNet trained on the labels generated by ours and the approach by Lottes et al. We also report the results when we use the uncertainty to post-process the semantic predictions. The bottom line shows a fully supervised approach trained on manual labels as upper bound of the performance. Best results per column in **bold**.

Approach	IoU [%]			mIoU [%]
	soil	crop	weed	
Lottes et al. + network	**99.1**	54.6	11.2	55.0
Lottes et al. + uncertainty	**99.1**	27.2	8.1	44.8
Ours + network	**99.1**	**88.8**	21.0	69.6
Ours + uncertainty	**99.1**	88.6	**22.7**	**70.1**
ERFNet + manual labels	98.0	83.4	33.5	71.6

shows the strongest boundary segmentation performance across all methods and classes on most datasets, often by a large margin compared to the second-best method. This further verifies our claim that our automatic labeling pipeline generates more accurate semantic soil-weed-crop labels than previous methods. We show qualitative results of Lottes et al. [132] and our approach in Fig. 5.6.

5.3.3 Unsupervised Semantic Segmentation

The second experiment evaluates the performance of our automatic label generation combined with network training and uncertainty post-processing. For that, we use the PhenoBench dataset. We show that training the evidential ERFNet using our automatically generated labels outperforms other unsupervised semantic segmentation models. The general-purpose learning-based approaches [77,161] have not been fine-tuned on human-labeled field images to ensure a fair comparison. We use our approach and the one by Lottes et al. [132] to generate labels for all images in the PhenoBench training set to train on the same set of images, thus having a fair comparison with the fully supervised ERFNet model trained on the manual labels.

Tab. 5.4 presents the performance of the network with and without uncertainty post-processing. We use "+ network" to refer to the results obtained by ERFNet after being trained on the labels generated by the approach, and we use "+ uncertainty" to refer to the results after we post-process them using the estimated uncertainty.

The results indicate that the approach by Lottes et al. [132] confuses more crops with weeds since it simply assigns all vegetation components that are not

on the main crop row to the weed class. This introduces inconsistent labels in the model's training data. Thus, training the ERFNet on Lottes et al.' labels does not yield uncertainty estimations that are useful for improving their predictions during post-processing. Additionally, this leads to smaller performance improvements after training on their labels than after training on our labels. Using our generated labels to train the ERFNet substantially improves the weed and crop predictions over directly using our generated labels. We further improve mIoU and weed predictions by exploiting the estimated uncertainties for post-processing of the network's predictions. Most importantly, combining our approach with the uncertainty refinement noticeably closes the performance gap between fully supervised and state-of-the-art unsupervised approaches. However, the ERFNet trained on human-labeled training images still predicts weeds more accurately. As the fully supervised model predicts more weeds, it also confuses weeds with crops more often. Hence, our approach performs better on both the crop and soil classes. This experiment confirms that our method's conservative labeling strategy, which excludes vegetation components prone to introducing labeling errors, combined with evidential deep learning, significantly reduces the reliance on manual annotations.

5.3.4 Generalization Capability

In the third experiment, we assess how our approach enhances the performance of networks trained in a fully supervised fashion by fine-tuning on unseen fields using our automatically generated labels. Instead of employing our full uncertainty-aware architecture, we fine-tune an ERFNet model using the standard cross-entropy loss pre-trained in a fully supervised fashion. We train two ERFNets, one on each of the human-labeled training sets of PhenoBench and SugarBeets. We deploy the two models on all the other datasets. Then, we fine-tune the two models, leveraging our automatically generated labels for the SugarBeets and PhenoBench datasets. Each model is fine-tuned on the dataset that it was not trained on.

In Tab. 5.5, we show the performance of the two models before and after fine-tuning. Due to the domain gap between datasets, models that were not fine-tuned have a lower performance when evaluated on unseen data. Fine-tuning alters the data distribution, leading the model to converge to a different local minimum. Usually, this deteriorates the performance on the original data, as the model prioritizes features relevant to the new distribution over those previously learned. Our results suggest that using our automatically generated labels helps to close the performance gap on previously unseen datasets with different crops, soil types, lighting conditions, and sensor setups. Generally, our fine-tuned models perform better on all classes and datasets, even on the Onions and Carrots

Table 5.5: Performance of fully supervised models trained on manually annotated data (first row), and fine-tuned on labels generated by our approach on a second dataset. In bold we highlight the best results per metric and per test set.

Train	Test	IoU [%]			mIoU [%]
		soil	crop	weed	
PhenoBench	SugarBeets	93.5	7.3	16.8	39.2
PhenoBench (+ SugarBeets)		**93.7**	**51.7**	**25.0**	**56.8**
PhenoBench	Carrots	**89.0**	11.1	**47.1**	49.1
PhenoBench (+ SugarBeets)		86.5	**26.0**	35.4	**49.3**
PhenoBench	Onions	82.4	0.5	**11.3**	31.4
PhenoBench (+ SugarBeets)		**87.7**	**5.5**	6.9	**33.4**
SugarBeets	PhenoBench	**97.6**	67.0	11.7	60.2
SugarBeets (+ PhenoBench)		97.5	**76.8**	**16.4**	**63.6**
SugarBeets	Carrots	87.6	36.1	24.3	49.0
SugarBeets (+ PhenoBench)		**88.6**	**38.2**	**34.3**	**53.7**
SugarBeets	Onions	86.3	0.2	13.2	33.2
SugarBeets (+ PhenoBench)		**87.3**	**12.3**	**13.9**	**37.8**

datasets that the model did not see during either the pre-training or fine-tuning. The model fine-tuned on SugarBeets does not gain much, i.e., less than 1% point, on the Carrots dataset. We hypothesize this is because the SugarBeets dataset is approx. 10× smaller than Phenobench, which introduces data imbalance, while also having lower quality labels compared to the manual annotations. In summary, using our automatically generated labels helps to fine-tune fully supervised models, enabling better adaptation to unseen field conditions without any additional human labeling costs.

5.4 Discussion

Most learning-based semantic segmentation approaches assume access to large amounts of human-labeled data required to train the vision model. However, their performance rapidly decreases in field conditions they were not trained on, i.e., different crop species, growth stages, weed pressure, and lighting conditions.

To address this issue, we proposed an automatic labeling approach to obtain semantic information from RGB images of agricultural fields. Our method has similar semantic segmentation performance to a fully supervised model trained on large amounts of human-labeled data. This significantly reduces the need for manually annotated data, diminishing costs and loosening the need for domain experts. The arable field datasets considered in our experimental evaluation report an average of 2 hours *per image* for labeling the Onions dataset, 3-4 hours *per image* for the Carrots dataset, and 1-3.5 hours *per image* for the PhenoBench dataset. All of the datasets went through at least two labeling rounds, doubling the costs. This suggests the need for new labeling paradigms besides fully supervised model training while maintaining strong prediction performance. Our method is a crucial step in narrowing the performance gap between models trained unsupervised and fully supervised models without additional labeling costs.

The results of our experiments indicate that the fully supervised approach has a lower performance in segmenting crops compared to our unsupervised method, as it is trained on more weeds. Nevertheless, the fully supervised method still shows the highest mIoU. The unsupervised methods are not exposed to enough weed labels, making them assign the crop class more often. Since the number of crop pixels is generally higher, these errors have a smaller impact on the crop than on the weed segmentation. We also investigate how to leverage our automatic labeling in combination with supervised methods to improve the overall performance in challenging scenarios, i.e., in unseen fields with new crop species and different weed pressure. Fine-tuning degrades performance on the pre-training dataset, as shown in Tab. 5.5. The degradation largely depends on the size and similarity of the pre-training and automatically labeled dataset used for fine-

tuning. Future work could investigate continuous learning methods to train on the newly automatically labeled images without catastrophically forgetting what has already been learned.

The need for images with associated pose information can be a limitation of our method, as it cannot be applied to a dataset of unposed images. However, most agricultural datasets are recorded using aerial or ground vehicles that, by default, provide spatial information while recording images in the field, often using GNSS systems such as GPS. Furthermore, we assume deployment in fields with a typical crop row structure. If this assumption does not hold, for instance in the presence of irregular planting patterns or when the weeds are larger than the crops, our crop row detection may fail, leading to degraded results. Addressing this limitation would require alternative unsupervised strategies that do not rely on the knowledge of the structured spatial arrangement. These could instead incorporate additional cues to distinguish between plant species, such as information from thermal or NIR images. Our results, along with those by Lottes et al. [132], demonstrate that a higher-quality vegetation mask could enhance the performance of unsupervised methods. One possible solution is to use NIR images, which are less dependent on the lighting conditions compared to RGB images. NIR images are already commonly used for crop segmentation in agriculture [44, 188]. Moreover, our approach leverages uncertainty estimates to post-process semantic predictions. Current state-of-the-art methods are known to produce partially miscalibrated uncertainty estimates [15]. Thus, our post-processing could benefit from improvements in uncertainty-aware deep learning. Finally, we plan to deploy our approach on a real robot to perform field trials.

5.5 Conclusion

In this chapter, we presented a novel approach to automatically generate semantic soil-crop-weed labels of images from agricultural fields. As a key aspect, our approach generates a semantic map of the entire field using robot pose information and exploits the row structure of the plant arrangement to produce high-quality and spatially consistent labels used for training semantic segmentation networks. We further enhance the semantic segmentation performance by employing evidential deep learning to estimate prediction uncertainties, enabling better identification of the typically under-represented weed class and allowing our targeted post-processing. Our results highlighted that incorporating domain knowledge, in this case about the spatial arrangement and the imbalanced distribution between crops and weeds, improves the quality of scene understanding tasks without the cost of additional labels. We evaluated our approach on four datasets recorded with different robotic platforms and in various fields. Our ap-

proach consistently outperforms previous domain-agnostic and domain-specific unsupervised labeling approaches. Moreover, we demonstrated that our generated labels enable fine-tuning fully supervised networks trained on one dataset for new agricultural fields, e.g., different species, growth stages, and field conditions, enhancing generalization capabilities without additional human labeling effort. Looking forward, the next critical step in field-based phenotyping requires us to distinguish individual plants, a challenge that we aim to address in the next chapter.

Chapter 6

Exploiting Crop Knowledge in Plant Instance Segmentation

Building on the semantic segmentation task discussed in the previous chapter, we now move to the next stage in the phenotyping pipeline: plant instance segmentation. While semantic segmentation classifies each pixel by class, i.e., soil, crop, or weed, instance segmentation goes one step further by assigning a unique identifier to every individual object in the scene. In essence, instance segmentation combines object detection and semantic segmentation by identifying each object in an image and delineating its precise shape at the pixel level. This is important in agricultural scenes, where separating individual instances allows for fine-grained analysis of fruits [99], plants [196,233], and leaves [74,137], laying the ground for trait extraction and growth monitoring.

In this chapter, we focus on plant instance segmentation using images of crop fields [3], as shown in Fig. 6.1. The goal of the task is to assign a unique ID to every observed plant. We target plant instance segmentation because it is a critical step towards phenotyping [30, 222]. It enables determining the growth stage of the crops, providing direct insights for yield estimation and targeted watering or fertilization, thereby reducing resource waste [152]. Plant instance segmentation is particularly challenging due to overlapping foliage and the irregular, complex shape of leaves. Heuristics- and learning-based approaches both struggle to address these problems.

The instance segmentation problem was originally tackled using heuristic-based techniques, which exploit geometric background knowledge about the domain. As we have already seen in this thesis, such knowledge can be useful for specific applications, and it can also be combined with learning-based systems to bootstrap approaches when training data is unavailable or hard to obtain [134, 181]. Today, machine learning, often deep-learning approaches such as convolutional neural networks or transformer architectures [41, 230] are widely

Figure 6.1: Robotic UAV systems equipped with RGB cameras, such as the drone shown in the left image, can capture images of agricultural fields for monitoring purposes. On the right we can see one exemplary image and the desired result for the plant instance segmentation task, where we assign a unique identifier to each plant. In the image, different identifiers correspond to different colors.

used. However, it is often required to adapt the heuristics or re-train the learning approaches to achieve satisfactory performance on a new crop species or field. This depends on the diversity of the crop varieties, on the common "closed world" assumption of the models that are trained on a small subset of classes, and on the ambiguous definition of crops and weeds in different agricultural settings, i.e., what is a crop in one field can be a weed in another.

Many approaches rely on neural networks trained on large datasets of paired texts and images, and a new paradigm has recently been proposed. These so-called vision-language models [121] have competitive performance compared to fully supervised methods in many computer vision zero-shot tasks, i.e., without adapting the model using additional training examples from the new domain. However, the performance deteriorates with increasing task complexity or when the application domain diverges too much from the original training dataset [210]. The domain gap is often tackled by fine-tuning the models on manually annotated data from the new application domain [79, 195]. We investigate a novel route to narrow the domain gap, leveraging domain-specific heuristics-based post-processing to enhance our final performance without requiring in-domain labels. Only a few works have explored the combination of heuristics- and deep learning-based approaches to get the best out of both worlds. This motivated us to develop

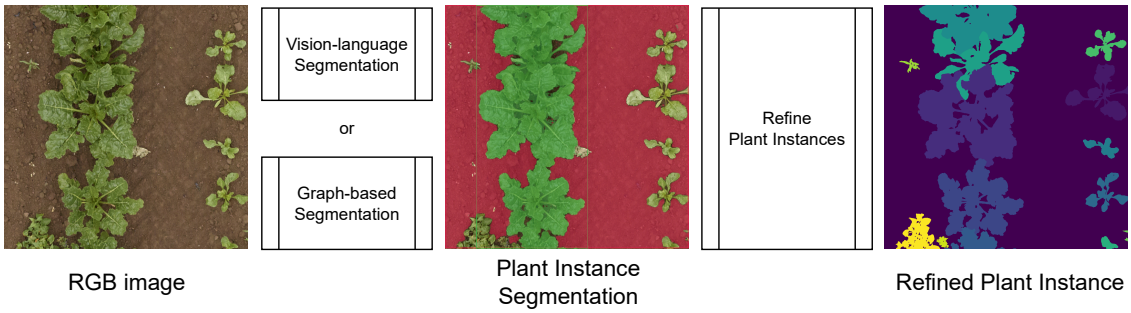


Figure 6.2: An overview of our approach. Taking as input an RGB image, we either perform a domain-specific graph-based segmentation or a vision-language model to obtain instance proposals. Then, we employ domain-specific heuristic to select which instances to split as explained in Sec. 6.1.3. The output is the unsupervised plant instance segmentation.

a pipeline to enhance deep-learning methods, i.e., reducing the need for labels and improving the networks' generalization capabilities. We illustrate our pipeline for an exemplary image in Fig. 6.2.

The main contribution presented in this chapter is a novel method to generate plant instance segmentation labels without requiring training data. We propose two ways to predict a first instance segmentation using (i) a vision-language model, or (ii) a graph-based image segmentation algorithm. These first predictions are then refined using domain-specific heuristic post-processing to improve our generated labels. Our experimental evaluation demonstrates that our approach can produce plant instance segmentation labels, which can then be used to train fully supervised methods, requiring fewer manually acquired labels and boosting their final performance. The experimental evaluation shows that our approach not only produces higher-quality plant instance labels compared to other state-of-the-art automatic labeling methods but also enhances the performance of neural networks when these labels are used as additional input. Moreover, our approach reduces the dependence on manually annotated data used as pre-training data, and it improves the model's ability to generalize to new, unseen fields where no ground-truth data is available.

6.1 Our Approach to Segment Plants Using Prior Knowledge

Our approach exploits domain-specific post-processing to refine the results of either a vision-language foundation model or a modified version of the graph-based image segmentation by Felzenszwalb et al. [62]. Given the set of detected objects, we refine the mask of every instance to ensure that it includes a single

plant, geometrically solving the problem of overlapping plants that confuses most approaches. Our refinement builds upon edge detection to determine the boundaries of the plants and includes domain knowledge to split the instances without requiring further training or additional manual labels. In the following sections, we will first explain vision-language foundation models in Sec. 6.1.1, and how we modified the approach by Felzenzwalb et al. [62] in Sec. 6.1.2. In Sec. 6.1.3, we discuss our heuristics-based refinement operation, while in Sec. 6.1.4 we introduce the optimization framework for the graph-based approach. Based on the proposed instance segmentation approach, we investigate multiple settings for training deep learning-based approaches in Sec. 6.2.3.

6.1.1 Vision-Language Model

Vision-language models comprise an object extractor that identifies objects in a given image using bounding boxes, and a segmentation network that generates pixel-wise masks for each detected object. Replacing the object detector or the segmentation network does not require any adaptation, since the two blocks have standard input-output pairs.

The object detector takes as input an RGB image $\mathbf{I} \in \mathbb{R}^{H \times W \times 3}$ and a set of text prompts. It extracts image features $\mathbf{X_I} \in \mathbb{R}^{N_I \times d}$ and text features $\mathbf{X_T} \in \mathbb{R}^{N_T \times d}$, where N_I is the number of image tokens, N_T the number of text tokens, and d corresponds to the feature dimension. These features are fused as $\mathbf{X} = \mathbf{X}_I \mathbf{X}_T^\top$ and then passed to a decoder to obtain the detected objects $o_i \in \mathcal{O}\, \forall i$, each one with its bounding box BB_i. The approach employs two thresholds: one to filter objects based on their prediction confidence, and one to filter them based on their alignment with given text prompts. Each object o_i is identified by a $prompt_i$ that reflects the object class, and a binary mask $\mathbf{M}_i \in \{0,1\}^{H \times W}$, where the value is 1 if that pixel belongs to the object, and 0 otherwise. To maximize the possibility of detecting all the relevant objects in the images we define $prompt_i \in \{\text{soil, crop, weed, single plant, vegetation}\}$, and assign to the same vegetation class all detections with $prompt_i \neq \text{soil}$. Using multiple synonyms for the vegetation class enables the model to capture the different vegetation components more accurately [72]. The filtered bounding boxes BB_i from the object detector are the input for the segmentation network to extract pixel-wise masks $\mathbf{M}_i$ for each bounding box. The mask has one associated "semantic class" which is the $prompt_i$ with the highest confidence score.

The results of such zero-shot instance segmentation models have two major short-comings: (i) the object detector allows some pixels in the image $\mathbf{I}$ not to be part of any detection; (ii) because of the difficulty of correctly separating overlapping plants, some detections need a refinement step to assign a unique ID to each plant. To solve the first problem, we compute the average RGB

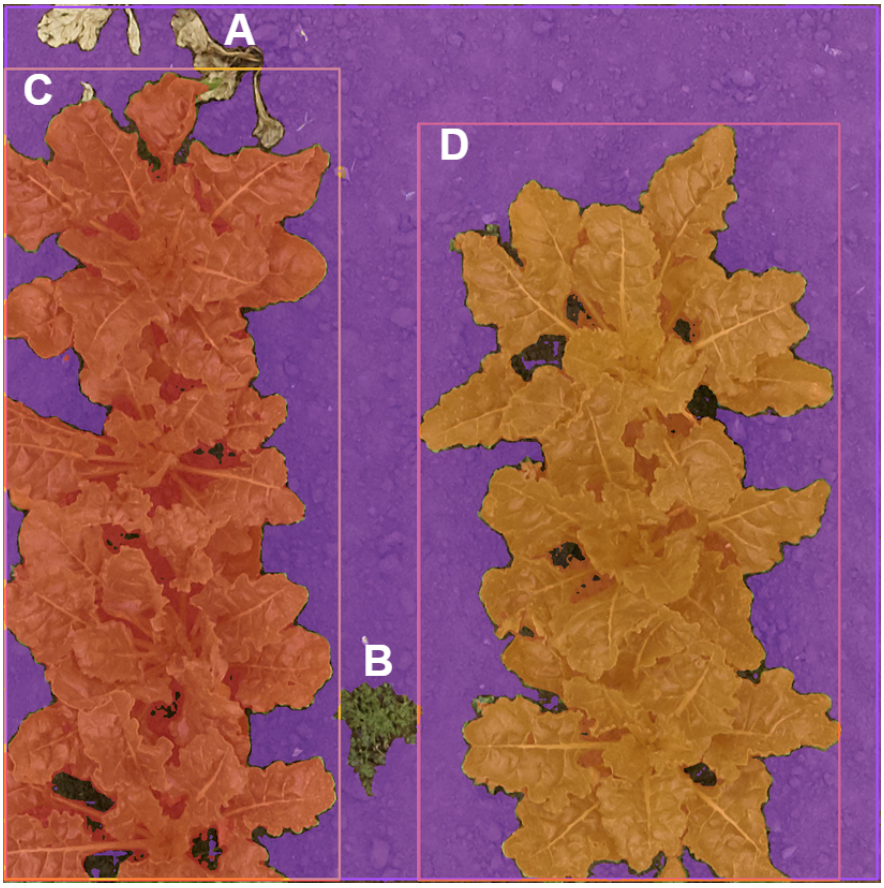


Figure 6.3: Output of Grounded SAM2, where soil is colored in purple and the vegetation instances are colored in different colors and surrounded by their bounding boxes. The leaves (A) in the upper-left corner are not segmented, as well as the weed (B) in between the two detected instances (C and D). Additionally, C and D both consist of multiple overlapping plants.

color for all pixels assigned to the vegetation and soil classes after the first step. We then use the cosine similarity to assign all not-segmented areas of $\mathbf{I}$ to the class – vegetation or soil – with the most similar color. Every area assigned to the vegetation class also gets a new instance ID. In this way, we correct both for undefined objects in the field, i.e., stones, wires, and pipes, that we want to assign to the soil class, and for missing vegetation detection. At the end of this step, all pixels have a semantic class, and every vegetation pixel is part of an instance. As already mentioned, to solve the second problem, we need a refinement step. Such a refinement operation is discussed in Sec. 6.1.3, as it is also required by the graph-based image segmentation method. We show in Fig. 6.3 both of these problems in an image from PhenoBench, where some leaves in the upper left corner (A) and a weed in the middle of the image (B) are not detected, and where multiple plants are segmented as a single instance (C and D).

6.1.2 Graph-Based Image Segmentation

The second possibility we explore to obtain candidates for the instances is the graph-based image segmentation by Felzenszwalb et al. [62]. We take as input the same RGB image $\mathbf{I}$, smooth it using a Gaussian kernel with standard deviation σ, and build for each image a graph $G = (V, E)$ with vertices V and edges E, where each pixel $\mathbf{p}_v$ is a vertex v, and each edge $e = (u, v) \in E$ between pixels u and v has a weight $w(u, v) \in \mathbb{R}$. We stick to the original implementation using the N_4 neighborhood, i.e., defining for each pixel of coordinates (i, j) edges with its four neighbors, with the coordinates $(i-1, j)$, $(i, j-1)$, $(i, j+1)$, and $(i+1, j)$. The weight $w(u, v)$ associated to each edge represents the dissimilarity between vertices v and u, i.e., the pixels $\mathbf{p}_v$ and $\mathbf{p}_u$. The dissimilarity is the measure used to define where to cut the graph and separate it into distinct segments.

The original implementation computes the dissimilarity $w(u, v)$, between u and v as

$$w(u,v) = \sqrt{(u_R - v_R)^2 + (u_G - v_G)^2 + (u_B - v_B)^2}, \tag{6.1}$$

where $(v_R,\ v_G,\ v_B)$ and $(u_R,\ u_G,\ u_B)$ are the values of the red, green, and blue channel of pixels $\mathbf{p}_v$ and $\mathbf{p}_u$ associated with vertex v and u, respectively.

In our work, we aim to represent only the differences between the ground and the vegetation; hence, we change the dissimilarity to

$$w(u,v) = \begin{cases} ||\psi(u) - \psi(v)||_2 & , \text{if } \psi(u) > \tau_{\text{ExG}} \text{ or } \psi(v) > \tau_{\text{ExG}} \\ 0 & , \text{otherwise} \end{cases}, \tag{6.2}$$

where $\psi(v) = 2v_G - v_R - v_B$ is the excess green index [228] computed for vertex v, and τ_{ExG} is a fixed threshold to compute the dissimilarity only for pixels that belong to the vegetation mask according to the excess green index. Once we have the graph, we initialize each node as a distinct segment in the image and run the algorithm by Felzenszwalb et al. [62] to combine the segments and obtain the final segmentation of the image. More specifically, the algorithm iterates over each edge $e \in E$, in increasing order of weights, i.e., from the edges between similar to dissimilar segments (pixels). Formally, two disjointed segments $\mathcal{C}_1, \mathcal{C}_2 \subset V$, where $\mathcal{C}_1 \cap \mathcal{C}_2 = \emptyset$, are merged together if

$$\min_{\substack{u \in \mathcal{C}_1, v \in \mathcal{C}_2 \\ (u,v) \in E}} w(u,v) \leq M \ \min(\beta(k, \mathcal{C}_1), \beta(k, \mathcal{C}_2)) \tag{6.3}$$

with

$$\beta(k, \mathcal{C}_i) = \max_{u', v' \in \mathcal{C}_i} w(u', v') + \frac{k}{|\mathcal{C}_i|}, \tag{6.4}$$

where M defines the margin of difference between the two segments and the maximum dissimilarity between two elements of the same segment to keep them separated, and k defines how fine or coarse the final segmentation should be. For further details on the implementation of the algorithm, we refer to the original paper by Felzenszwalb et al. [62].

At this point, we have an instance segmentation with no semantic information. We use the mean of the RGB values for each segment $\mathcal{C}_i$ to distinguish between soil and vegetation. We use the excess green index and the same threshold τ from Eq. (6.2) to assign the semantic classes y, as

$$y(\mathcal{C}_i) = \begin{cases} \text{soil} & \text{, if } \frac{1}{|\mathcal{C}_i|} \sum_{u \in \mathcal{C}_i} \psi(u) < \tau_{\text{ExG}} \\ \text{plant} & \text{, otherwise} \end{cases}. \tag{6.5}$$

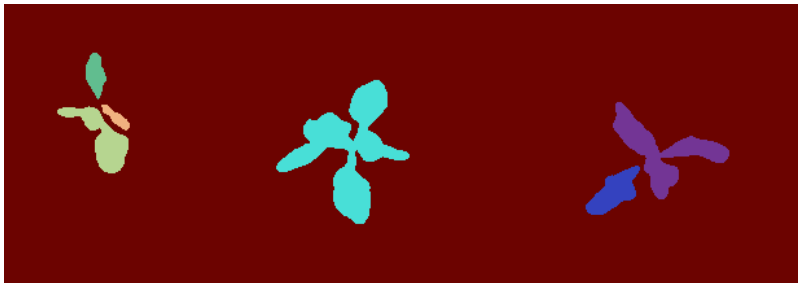
Using the graph-based image segmentation method has one advantage over using VLMs: we always obtain a class for all pixels in the image. However, since the classification is based on heuristics and not learned from data, it is possible that the chosen threshold τ_{ExG} does not provide a correct prediction when there is a gap between training and testing data. Such situations are common in the agricultural domain when we encounter changes in light, soil color and texture, and crop species.

To obtain the instances, we need to ensure that we merge the leaves that have been separated from the main body of the plant into one component. The over-segmentation of a single plant can happen because of lighting conditions, small peduncles that are not visible after smoothing the image, and boundaries between adjacent and overlapping leaves. To better follow the next steps of the approach, we visualize them in Fig. 6.4. In Fig. 6.4 (a) we see the graph segmentation, and in Fig. 6.4 (b) the output of Eq. (6.5). We then merge close segments of the vegetation using dilation. For a pixel $\mathbf{p}$ of coordinates (i, j), the dilation returns a value depending on its neighbors $\mathcal{N}(i, j)$ as

$$\mathbf{p}(i, j) = \max_{\mathbf{p}(x,j) \in \mathcal{N}(i,j)} \mathbf{p}(i + x, j + y). \tag{6.6}$$

This operation can be applied n times iteratively and has no defined concept of neighbors. In our implementation, we use $\mathcal{N}(i, j) = \{(r, s) \mid -\gamma \leq r, s \leq \gamma\}$, where the kernel size γ defines the size of our neighborhood. The kernel size γ and the number of iterations n are both hyperparameters we can tune. The result of the dilation is shown in Fig. 6.4 (c). After dilation, we can compute the connected components [22] as depicted in Fig. 6.4 (d). A connected component in a binary image is a set of adjacent pixels that share the same property: in our case, there will be components with value 0 or 1.

At this point, we have an instance segmentation, but we need to filter out the pixels that were not in the original vegetation segments. We need to multiply



(a) Graph segmentation

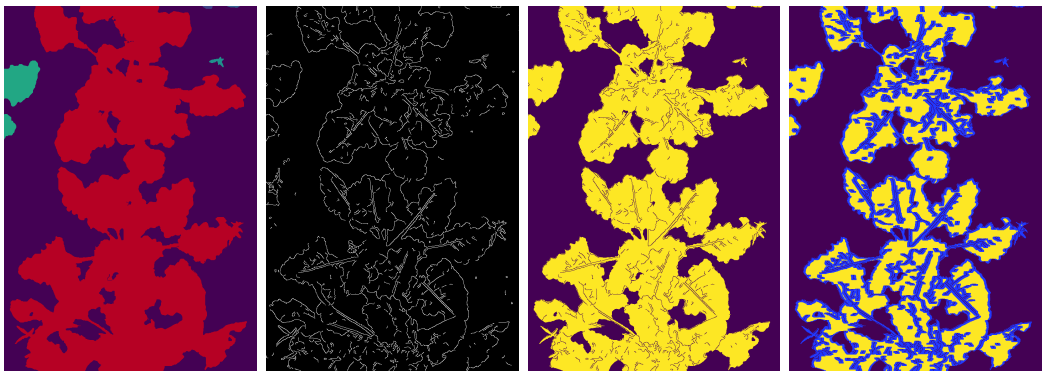
(b) Extracted vegetation mask

(c) Dilated mask

(d) Connected Components

(e) Product of (d) and (b)

Figure 6.4: From top to bottom we show the graph segmentation, the extracted vegetation mask, the mask after dilation, the result of the connected components, and the product of the connected components with the vegetation mask.

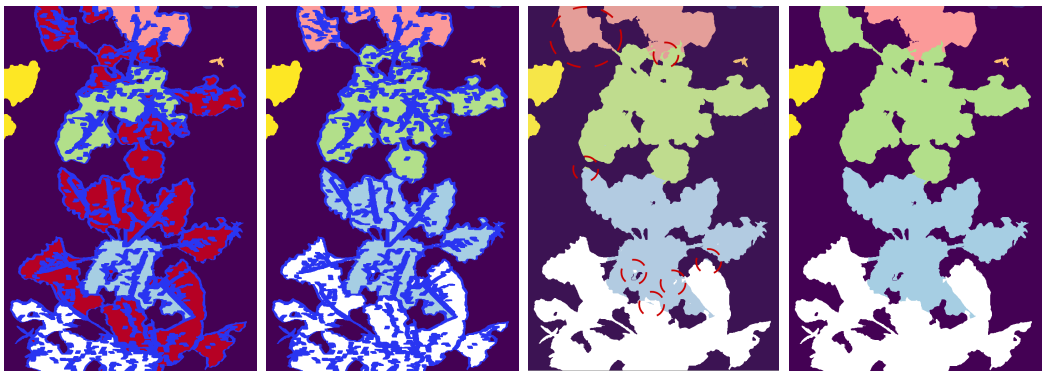


(a) Instance before refining

(b) I_{edges}

(c) $I_{e+\texttt{inst}}$

(d) $I_{e+\texttt{inst}}$ after erosion



(e) Largest $N_{\texttt{inst}}$ components

(f) Assignment of all components

(g) Final points' assignment

(h) Mask after cleaning

Figure 6.5: Step-by-step images depicting how we address the splitting of instances. In (a), we see the unified instance in red, while in (b) we show its edges. (c) shows the results of the XOR operation between the instance mask and I_{edges}. In (d), we can see the result of the erosion; eroded points are colored in blue. In (e), we show the largest components colored in white, azure, green, and pink, while in (f) we see the assignment of all the other components. After assigning the blue points using a voting mechanism, in (g), we use red dotted circles to highlight points assigned to one instance but not connected to it. In (h), we show the final instance segmentation after cleaning.

again for the semantic mask, filtering out all the pixels originally classified as soil. The output of this step is the plant instance segmentation shown in Fig. 6.4 (e), i.e., one image where each pixel has either value 0 if it is soil or a positive value N as the instance ID of the predicted plant.

As for the results of the vision-language model, we are left with the problem of separating instances of overlapping plants that are currently merged.

6.1.3 Plant Instance Segmentation Refinement

The methods used to produce candidates for the plant instance segmentation have difficulties separating single plants when they overlap. We propose refining these instances by once again leveraging domain knowledge.

First, we need to detect which instances to refine. Using crop-specific knowledge, we can design a split function $f(\mathbf{I}_{\text{inst}}) \in \{\texttt{True}, \texttt{False}\}$ that takes as input one instance binary mask $\mathbf{M}_i$ and return `True` if the instance needs to be split. In our implementation, we use the aspect ratio $a = \frac{H_i}{W_i}$ to detect if the instance needs to be refined. In this way, we avoid using the number of pixels as a proxy for when to split the instance, which is problematic when we encounter plants of different growth stages, i.e., the same number of pixels could belong to two early growth stage plants or one late growth stage plant. Thus, we define the split function as follows:

$$f(\mathbf{M}_i) = \begin{cases} \texttt{True} & \text{, if } a > \tau_a \\ \texttt{False} & \text{, otherwise} \end{cases}, \tag{6.7}$$

where τ_a is the aspect ratio threshold. At this point, we can also have an estimate of how many instances N_i have been aggregated according to our threshold, as

$$N_i = \lceil a/\tau_a \rceil. \tag{6.8}$$

Eq. (6.8) can be expanded as $N_i = \lceil \frac{H_i}{W_i} \frac{W_\tau}{H_\tau} \rceil$. We use the aspect ratio because it is independent of the plant size and image resolution. Since the number of instances can only be an integer, we only take the integer part of the result, which also provides a tolerance, i.e., an aspect ratio $a = 1.3$ with a threshold $\tau_a = 1$ detects only one instance with a margin of 0.3 of difference in the ratio. We do not allow the number of instances to be less than one. If this assumption is violated, we compute the aspect ratio as W_i/H_i, which means that the crop row in our image is oriented horizontally instead of vertically.

We refer to Fig. 6.5 to illustrate the steps we perform once we identified an instance that needs to be split, since it actually comprises multiple plants. Fig. 6.5 (a) shows one red instance that our approach decides to split. We can compute the edges $\mathbf{I}_{\text{edges}} \in \{0, 1\}^{H \times W}$ from the original RGB image with any edge

detector: in our implementation we first apply a smoothing to the image and then use the edge detector by Canny et al. [25]. $\mathbf{I}_{\text{edges}}$ is shown in Fig. 6.5 (b). We then exclude the edges from the instance mask with a bit-wise XOR operation between $\mathbf{I}_{\text{edges}}$ and $\mathbf{M}_i$, we call the output $\mathbf{I}_{\text{e+inst}}$ shown in Fig. 6.5 (c). Since we cannot guarantee that the edge detector finds smooth and optimal edges to separate our instances, we erode $\mathbf{I}_{\text{e+inst}}$ using a kernel of size γ; this will expand our edges and better separate the instance, as can be seen in Fig. 6.5 (d). We apply connected components and select the N_i components with the largest areas as our new instances. The N_i instances detected in our example are the ones colored in white, light blue, green, and pink in Fig. 6.5 (e). We iteratively assign all the other components to the closest instance, computing the Euclidean distances between the centers of the components. The result of this iterative process is shown in Fig. 6.5 (f).

Now, we need to take care of pixels belonging to the original instance and that were removed with the erosion, depicted in blue in Fig. 6.5 (f). For each of them, we compute the set of neighbor pixels belonging to an instance. We then count the number of neighboring pixels assigned to each instance and assign the pixel to the instance with the highest count. The result is shown in Fig. 6.5 (g). There, we see in red dotted circles the pixels assigned to an instance but not connected to it. We clean the mask, keeping the largest component for each instance ID as it is, while assigning the smaller components to an existing instance connected to it, or a new instance otherwise. The refined instances after these operations are shown in Fig. 6.5 (h).

6.1.4 Graph-Based Hyperparameters Optimization

When we use the graph-based image segmentation method, our approach is entirely based on heuristics and has seven hyperparameters. Three are for the graph-based segmentation: the kernel for the Gaussian smoothing σ, the margin between the maximum internal and external dissimilarity M, and the coarseness of the segmentation k. The additional four parameters are the threshold for the excess green index τ_{ExG}, the kernel size γ, the number of iterations n for the dilation, and the aspect ratio threshold τ_a for the post-processing and refinement.

We use Optuna [7] to perform hyperparameter optimization, aiming to maximize the Panoptic Quality [108] over the images of the training set. We fix the excess green index threshold $\tau_{\text{ExG}} = 0.2$ and the aspect ratio threshold $\tau_a = 1$. We found these values by inspecting different images and their mean green excess index masks. We validate our choice for the value of τ_{ExG} by computing the excess green index value over vegetation pixels of different datasets. All results were between 0.2 and 0.25. The IoU of our approach also suggests that the threshold can capture the vegetation components of different datasets. We fix the kernel

size $\gamma = 3$, which is the default value in most of the graphic libraries [22, 212], and the number of iterations $n = 10$, which for high-resolution images is a 2 cm dilation, i.e., the maximum error we allow in our vegetation mask to reconnect components of the same plant. We use these values for all experiments except those in which we estimate the importance and range of the hyperparameters, where we show that these values are in the optimal ranges.

This leaves us with only three parameters to fine-tune, which are those from the original implementation of the algorithm: the kernel for the Gaussian smoothing σ, the margin between the maximum internal and external dissimilarity M, and the coarseness of the segmentation k. For each hyperparameter to tune, we must define its type, i.e., integer or floating point, and the range of possible values. The framework runs the same experiment multiple times, varying the hyperparameters based on the results of the previous trials. In our case, each trial returns the panoptic quality, and we seek to maximize its value. We use the tree-structured Parzen estimator [165] to sample the hyperparameters.

Given the number of parameters to optimize and their range, we can compute the number of trials T required by Optuna [7] to try every combination. If $\mathcal{H}$ is the set of hyperparameters and range(h) is a function returning the number of possible values of h, we can compute

$$T = \prod_{h \in \mathcal{H}} \text{range}(h). \tag{6.9}$$

With only three hyperparameters to optimize, we can make fewer trials to find the best configuration, focusing on the optimization of the graph hyperparameters. Importantly, hyperparameter tuning is required only to obtain the best possible performance. As we will show in the experiments, this step is optional as our approach can generalize across different datasets, growth stages, and illumination changes with a fixed set of hyperparameters.

6.2 Experimental Evaluation

The main focus of this work is a fully unsupervised pipeline for plant instance segmentation that exploits vision-language foundation models or graph-based image segmentation and domain-specific post-processing. The approach takes RGB images as input and computes plant instance annotations that we use to (i) boost the performance of networks on data for which we have labels and (ii) improve the generalization of a network on different fields.

In Sec. 6.2.2, we show the results of different vision-language and heuristics-based methods and how our domain-specific post-processing improves their results on different agricultural datasets; then in Sec. 6.2.3, we show how to use

our generated labels to improve the generalization capabilities and reduce the requirement for manually annotated data of fully supervised learning methods.

6.2.1 Experimental Setup

Datasets. We test our approach on three RGB agricultural datasets. Two of them are recorded on fields of sugar beets: one was introduced by Weyler et al. [222], denoted as SugarBeets, and the other is the public benchmark dataset PhenoBench [224]. The third dataset is GrowliFlower [106], which is recorded on a field of cauliflowers. The three datasets have different image resolutions, lighting conditions, and growth stages; furthermore, only PhenoBench provides weed annotations. We use the official validation and test sets for all of them. The SugarBeets dataset, which is not publicly available, consists of 745 images for training, 272 images for validation, and 278 images for testing.

Metrics. We compute the intersection-over-union (IoU) [60] for the vegetation, or as the mean over crops and weeds. We also compute the panoptic quality (PQ) [109] to evaluate the quality of the instance segmentation.

Details and hyperparameters. Our approach with vision-language models has two hyperparameters: the kernel size γ and the aspect ratio threshold τ_a; while our approach with the graph-based image segmentation method has seven hyperparameters, of which we tune three using the Optuna optimization framework [7] as already mentioned in Sec. 6.1.4. We analyze the importance and the range of the optimal hyperparameters in Sec. 6.2.4.2. In all the experiments, we fix the aspect ratio threshold $\tau_a = 1$ and the kernel size $\gamma = 3$. We train all networks using the configuration suggested in their original papers unless they give different parameters for the specific dataset.

Baselines. We benchmark against heuristic-based approaches similar to our domain-specific post-processing and the results of the vision-language models without our post-processing. In particular, we try two different object detectors, Grounding DINO [128] and Florence2 [229], and two versions of the Segment Anything Model [175], SAM2 and SAM2.1. These changes do not alter the input that we provide or the outputs that the foundation models supply to our domain-specific post-processing. Detailed information about the different object detectors and pipelines can be found in the original papers.

As a first heuristics-based baseline, we use the original implementation of the graph-based image segmentation by Felzenswalb et al. [62]. The comparison to this baseline helps us assess how much our domain-specific adaptations improve the performance of the segmentation compared to the general-purpose method. The second heuristics-based baseline is the vegetation mask based on the hue histograms [81], which is a commonly used option that does not suffer from the changes in lighting and weather conditions affecting the RGB values of the images.

Table 6.1: Results of the vegetation IoU and PQ for all baselines on all the different datasets. In bold the best results for each metric and dataset. Our results highlighted in gray. All results are given in %.

	Approach	SugarBeets		GrowliFlower		PhenoBench	
		IoU [%]	PQ [%]	IoU [%]	PQ [%]	IoU [%]	PQ [%]
heuristic	Graph-based [62]	58.1	47.8	62.7	35.2	68.3	3.9
	HUE [81]	67.8	34.8	71.3	13.9	74.5	2.6
	ExG [228]	73.1	66.8	76.3	24.5	75.1	22.6
	Graph-based + ours	**76.6**	70.1	84.1	75.9	**81.8**	51.4
VLMs	Grounded SAM2	72.9	78.6	72.0	74.1	58.2	60.6
	Grounded SAM2 + ours	75.2	78.1	88.1	79.0	77.3	66.3
	Florence2 + SAM2	33.4	47.5	78.3	61.3	59.6	44.2
	Florence2 + SAM2 + ours	72.2	75.4	80.9	82.9	62.9	**67.0**
	Grounded SAM2.1	69.9	**86.3**	66.4	84.0	45.3	62.7
	Grounded SAM2.1 + ours	75.1	83.3	**88.6**	**85.2**	78.7	66.0

Since we use the excess green index in our dissimilarity function, we report the results of the excess green index [228] where we employ a threshold to obtain a vegetation mask based on the predominance of the green color in the vegetation. For all the heuristics-based baselines, we compute the plant instance segmentation from the vegetation masks as explained in Sec. 6.1.2.

We use the same deep-learning baselines we compared to in Chapter 3, retaining their acronyms. Mask R-CNN [83], denoted as MR, PanopticDeepLab [39] with MobileNetV2 [191] as the backbone, denoted as PD-S, and the approach by Weyler et al. [222], denoted as Weyler. As a last deep-learning baseline, we use our approach introduced in the same Chapter 3, here denoted as HAPT.

6.2.2 Experiments on Unsupervised Label Generation

In this section, we evaluate the performance of our two approaches, i.e., the one using the graph-based image segmentation and the one using the predictions of the vision-language model. The experiments indicate that our methods outperform common heuristics-based approaches in terms of segmenting soil and vegetation, as well as in distinguishing individual vegetation instances, while improving the performance of the foundation models.

Tab. 6.1 shows the results on all three datasets for all baselines. On the SugarBeets dataset, the VLMs and heuristics-based methods all have similar IoU results, except for Florence2, which produces fragmented masks. The other VLMs have a better PQ than all heuristic-based methods. The version of our approach based on graph image segmentation is the best among the heuristic methods. The use of a graph-cutting method instead of simple thresholding makes our method

Figure 6.6: Qualitative images from Grounded SAM2 (left), and Grounded SAM2 + our post-processing (right). We highlight in red dotted circles the errors by the two approaches, and in green the correct prediction.

more robust than the baselines when performing instance segmentation. Incorporating domain knowledge into the graph-based segmentation enhances both metrics across all datasets, yielding an average improvement of 17.8 percentage points for IoU and 37.1 for PQ. Adding our post-processing to the VLMs improves the IoU in all investigated cases, but worsens the PQ for the models based on Grounding DINO, i.e., Grounded SAM2 and Grounded SAM2.1. We investigate this further by looking at qualitative results. In the image shown in Fig. 6.6 Grounded SAM2 had a vegetation IoU of 61.7% and a PQ of 95% since it only missed the plant in the red dotted circle. After our post-processing, the IoU is 75.5% because we are correctly identifying the missing plant as vegetation, but we also classify the weed at the bottom as vegetation. This error brings our PQ to 87.5%. Since in the ground truth all weeds are labeled as soil, our correct vegetation detection for weeds is considered an error. This usually has a higher impact on the PQ than on the IoU because, although the ratio between misclassified pixels and vegetation pixels is low, the number of incorrect detections is high compared to the number of objects in the image.

For the GrowliFlower dataset, we show in Tab. 6.1 that all the approaches yield good performance in differentiating vegetation and soil, probably thanks to good lighting conditions. However, the presence of various growth stages makes the instance segmentation task harder. The version of our approach based on graph image segmentation outperforms all heuristic approaches and part of the VLMs without our additional post-processing. GrowliFlower has some images with grass that should be detected as soil, as it is not a crop to harvest or a weed to remove. It is hard for VLMs and heuristics-based methods alike to classify grass as soil. The VLMs have similar performance to the heuristic-based approaches for the IoU but superior results in the PQ. Using our domain-specific post-processing improves the results of all VLMs, both in IoU and PQ. The two most impressive results are the IoU of Grounded SAM2.1, improved by 22.2 percentage points, and the PQ of Florence2 + SAM2, improved by 21.6 points.

These improvements show that we correct both for missing vegetation detection and for wrongly merged instances.

The rightmost column of Tab. 6.1 shows the results on PhenoBench, the only dataset with weed annotations. The dataset presents images from late growth stages when many leaves overlap and create shadows, making the segmentation task challenging for both heuristics- and neural network-based approaches. Without being penalized for detecting the weeds, all heuristic approaches achieve higher IoU compared to their performance on the SugarBeets dataset (average of +7%). However, the more challenging environment produces a loss in terms of PQ (average of -23.6%). Our version of the graph-based image segmentation is, among the heuristic approaches, the one that gets the second greatest increase in IoU, and the smallest decrease in PQ. The presence of shadows, weeds, and multiple growth stages limits the semantic segmentation of VLMs, all attaining lower IoU than heuristic-based methods. However, the VLMs have superior abilities in differentiating single plant instances, even with their reduced set of correct vegetation pixels. Again, adding our domain-specific post-processing improves the IoU and PQ of all methods, surpassing most heuristic-based methods in terms of IoU and boosting the quality of plant instance segmentation.

6.2.3 Experiments on Exploiting Our Generated Plant Instance Labels

In this section, we investigate several scenarios where we employ our approach to boost the performance of deep-learning approaches on the plant instance segmentation task. The results illustrate that our approach (i) boosts the performance of neural networks when used as additional input; (ii) reduces the need for labels when used as ground-truth annotation; (iii) improves the performance on plant instance segmentation when used to generate extra labels for the training procedure; (iv) helps the network generalize better on different fields without the need for ground-truth annotations.

6.2.3.1 Generated Instances as Additional Input

This set of experiments demonstrates that, even when we have access to labeled data, our heuristic approach can be used to improve the performance of learning-based systems. We conduct two types of experiments. Firstly, we augment the input of the networks with our generated labels, secondly, we add the offset vectors for each instance we detected. We evaluate this experiment on all three datasets and with all the learning-based approaches. We run all experiments under the same configuration and with a fixed random seed so that the only change is in the additional inputs provided.

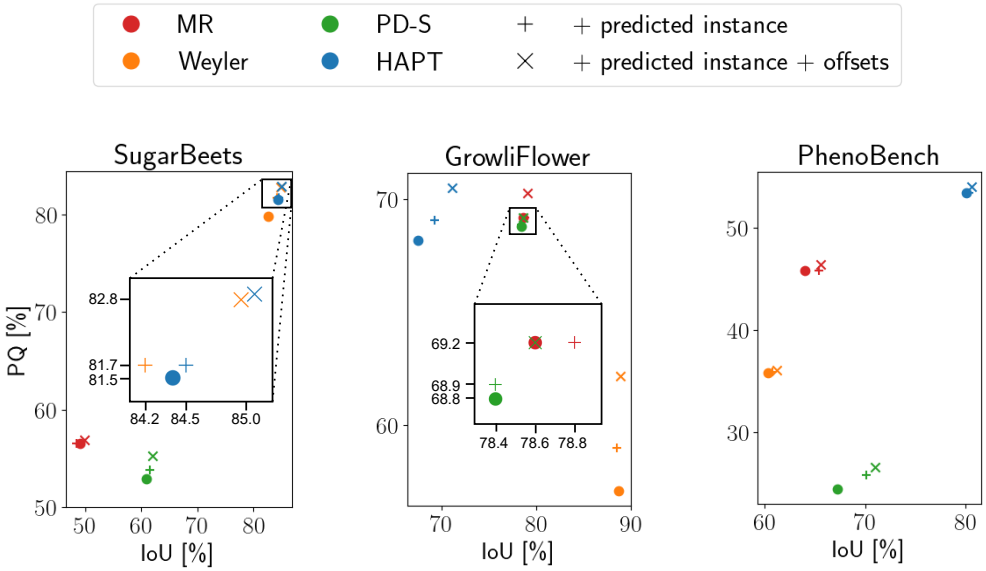


Figure 6.7: We show the results of the deep-learning networks (o) against the results obtained by the same network when augmenting the input with our predicted instances (+), or with the predicted instances and the offsets computed from them in the x and y direction (x).

Fig. 6.7 shows the results of the experiments on the three different datasets. Using the additional inputs helps the network and boosts the final performance on both PQ and IoU for all experiments. The only cases in which this is not true can be easily explained. For the SugarBeets dataset, MR using only the generated labels as additional input has a lower IoU. As this dataset has no weed annotation and our approach, without semantics, provides instances for each vegetation component, i.e. weeds and crops, the network does not learn to ignore the weeds, which are considered an error during the evaluation.

For the GrowliFlower dataset, only the approach by Weyler et al. [222] has a lower IoU when using our labels as additional input. The reason is similar to the one given for MR on the SugarBeets dataset, since Weyler does not learn any semantics, it cannot learn to ignore the weed instances. As an additional point, even if the metrics are correlated, it can happen that the model with the best PQ is not the model with the best IoU. This occurs when the model predicts cleaner instances, improving the PQ, while underestimating the object extent, resulting in lower pixel-level overlap. In this case, the maximum IoU obtained by the model is 1.7 percentage points higher than the one reported in Fig. 6.7, the same as the baseline without additional inputs.

For the PhenoBench dataset, all approaches have improved their IoU and PQ when using our additional inputs. Since this dataset presents both crop and weed annotations, the IoU reported in Fig. 6.7 is the mean IoU over all classes.

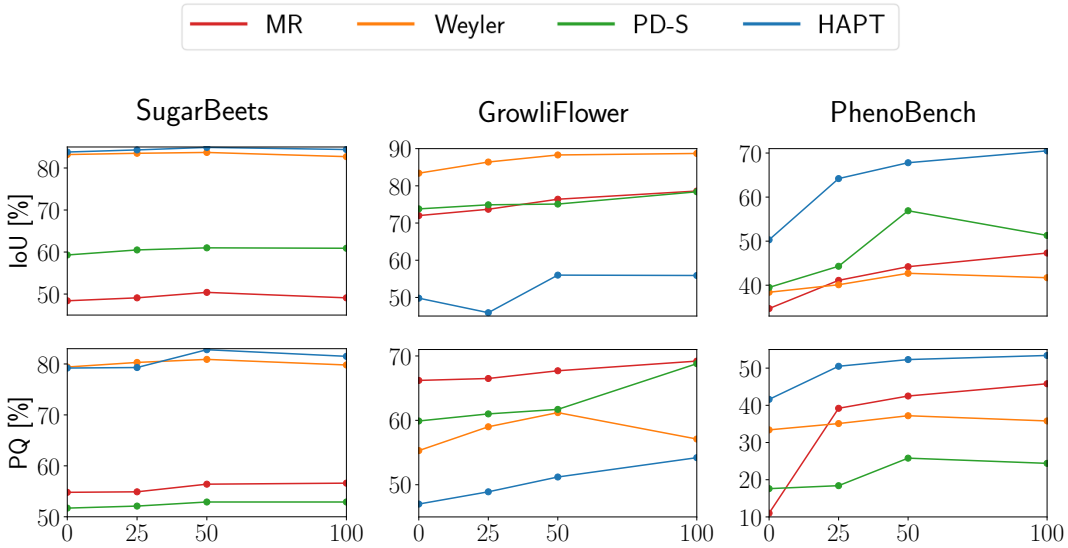


Figure 6.8: Results of three deep-learning networks, when using our approach instead of manual labels. On the x-axis we show the percentage of manual labels used during training. The first row depicts the IoU, i.e., crop on SugarBeets and GrowliFlower, and the mean of crop and weeds for PhenoBench. The second row shows the PQ.

We note that for this dataset, the additional inputs are helping the IoU more than the PQ. We believe this is because the PhenoBench dataset consists of high-resolution images $\left(1\frac{mm}{px}\right)$, and it presents small plants which are hard to detect for convolutional networks with big receptive fields. In the case of the graph-based image segmentation version of our approach, we do not perform any downsampling, thus enhancing the visibility of small instances for the network. Similarly, the refinement applied to the predictions of the VLMs to obtain pixel-wise semantic classes also operates on the image at its original resolution.

6.2.3.2 Labels Substitution

This experiment aims to show the capability of our approach to reduce the need for human-generated labels. For all datasets and baselines, we run three experiments, progressively substituting the manually annotated labels with the output of our pipeline. We run the experiments substituting 50%, 75%, and then 100% of the human-generated labels. The approach by Weyler et al. [222] is a bottom-up method that requires leaf instance labels as supervision, thus it could infer the manual plant labels from the provided leaf instances. We nevertheless decided to conduct the experiments using this approach by substituting only the plant instances with our approach and analyzing the results.

In Fig. 6.8, we see the results of the experiments. For SugarBeets and GrowliFlower, which both only have crop and soil as semantic classes, the network can learn the task and perform well on the test set using less than half of the

labeled data. The metrics are slightly lower than those obtained from training on all the labeled data, but the drop in performance of 2 percentage points is a good compromise if we need to label only 1/4 of the images. Interestingly, some of the experiments have better metrics when using only part of the real labels. This effect is not consistent across datasets or approaches: we can observe it in the IoU of MR for SugarBeets, in the IoU of HAPT, in the PQ of Weyler for the GrowliFlower dataset, and in the metrics of PD-S on PhenoBench. We visually investigated these results and concluded that, considering the performance shown in Tab. 6.1, the error introduced by our labels is considered part of the data noise when there are enough manual labels to drive the learning-based approaches in the right direction. For the SugarBeets and GrowliFlower datasets, our generated labels are likely to have weed instances that can be considered hard negatives for the network to better learn the final task, leading to a small improvement.

6.2.3.3 Additional Labels

In these experiments, we aim to assess whether scaling up the number of images in our training data by including images annotated by our approach can boost the performance. We evaluate this capability by testing the models on a joint test set comprising the validation set from PhenoBench and the test set from GrowliFlower. We create different training datasets, consisting of manually labeled data from PhenoBench and images from other datasets labeled with our approach. We then evaluate how this diversity helps or degrades performance on both datasets. Using our generated plant instance labels for the "Extra data", we can provide semantic annotations only in terms of vegetation and soil.

First, we generate plant instance labels for an additional sugar beet dataset introduced by Ahmadi et al. [4], which contains 287 images. This dataset has different lighting conditions, growth stages, and image resolution compared to PhenoBench, but it presents the same crop species. The ability to adapt to new scenarios, even when the objects in the scenes are the same, is part of what domain adaptation algorithms try to solve. In the second experiment, we further shift the domain by using a corn dataset consisting of 280 images introduced by Ahmadi et al. [5]. Corn is not in our test set, containing PhenoBench and GrowliFlower, i.e., sugar beets and cauliflowers. Nevertheless, we believe that a network can benefit from seeing different crop species, as our ultimate goal is to use the same network for any crop species, seen or unseen. In the third experiment, we extend the training set with all the $1,542$ images from the original GrowliFlower training set, using our labels. In this case, the data presented at training time has a more similar distribution to the test data, i.e., sugar beets and cauliflowers. The number of new images is comparable to the size of the original training set, so the networks should be able to optimize equally for both crop species. We run all

Table 6.2: Results on the validation sets of GrowliFlower and PhenoBench, both independently and together, for MR. We train on PhenoBench only and with additional labels provided by our approach on different datasets.

Extra Data	Test Set		IoU [%]			PQ [%]
	PB	GF	soil	crop	weeds	
none	✓		**97.3**	**70.9**	23.7	45.8
		✓	76.8	9.0	-	7.9
	✓	✓	90.5	50.2	23.7	32.8
Sugar Beets [4]	✓		96.8	66.7	39.9	**46.8**
		✓	78.0	15.9	-	10.8
	✓	✓	90.5	49.8	39.9	34.8
Corn [5]	✓		96.9	63.2	**40.0**	45.5
		✓	**86.7**	**34.3**	-	30.9
	✓	✓	**93.5**	53.6	**40.0**	**40.6**
GF (Train) [106]	✓		96.2	67.5	30.5	33.9
		✓	81.4	32.7	-	**34.1**
	✓	✓	91.3	**56.0**	30.5	34.0

experiments using Mask R-CNN, PanopticDeepLab, and HAPT and report the metrics in Tab. 6.2, Tab. 6.3, and Tab. 6.4. We cannot run the experiments on Weyler et al.'s approach since it requries supervision from the leaf instances, and we cannot provide them using our pipeline.

We can see that, in general, introducing the additional data always has a good impact on the PQ of the combined evaluation set and on the PQ and IoU of GrowliFlower. For Mask R-CNN and HAPT, several metrics diminish on the source domain. This is expected since the weights need to be optimized for new crop species, growth stages, and field conditions. PanopticDeepLab is the architecture that benefits the most from the additional data. This depends on its use of centers and offsets instead of the region proposal of Mask R-CNN, and on the size of the network, which is three times larger than HAPT, making the network less prone to overfitting to the training data. Looking for the best results, we can see that we obtain most of them using GrowliFlower or Corn as additional data; the first is expected since the training distribution matches the one for evaluation, while the second suggests that using different crop species can increment the ability of the networks to generalize even if the new species are not presents in the final evaluation data.

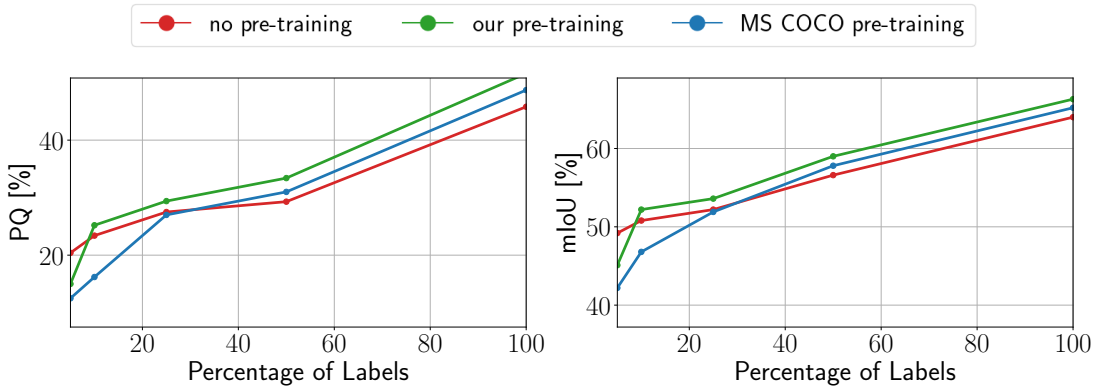
Table 6.3: Results on the validation sets of GrowliFlower and PhenoBench, both independently and together, for PD-S. We train on PhenoBench only and with additional labels provided by our approach on different datasets.

Extra Data	Test Set		IoU [%]			PQ [%]
	PB	GF	soil	crop	weeds	
none	✓		99.0	81.5	21.0	24.4
		✓	78.4	35.5	-	30.8
	✓	✓	92.1	66.2	21.0	26.5
Sugar Beets [4]	✓		97.4	80.3	22.6	31.5
		✓	89.8	65.4	-	40.3
	✓	✓	94.9	75.3	22.6	34.4
Corn [5]	✓		**99.1**	**84.7**	**29.1**	**36.4**
		✓	94.0	78.5	-	45.8
	✓	✓	97.4	83.6	**29.1**	39.5
GF (Train) [106]	✓		99	84.1	27.8	33.7
		✓	**95.1**	**86.5**	-	**58.6**
	✓	✓	**97.7**	**84.9**	27.8	**42.0**

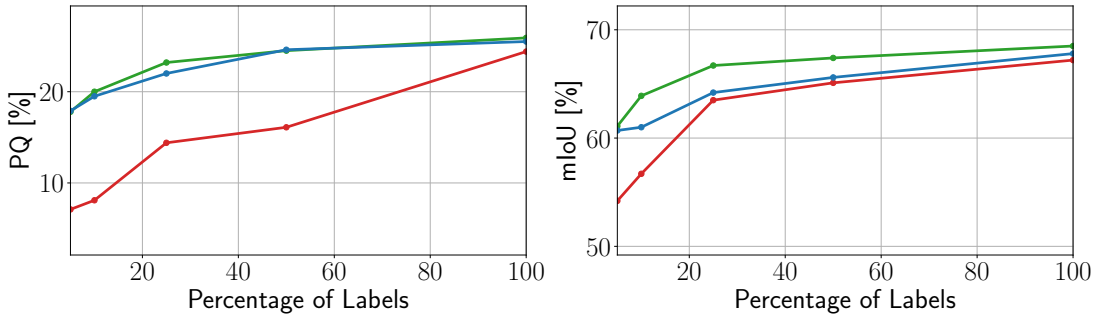
6.2.3.4 Pre-Training

This experiment aims to show that our plant instance labels can be used to pre-train any backbone for the plant instance segmentation task. Pre-training is a common strategy to initialize the weights of a neural network, reducing the amount of data and iterations to converge to the optimum [58, 202]. Our pre-training, in this case, can be considered self-supervised since our labels are generated without training on labeled data. However, compared to common pre-training strategies, in which the pre-training and final tasks are related but not identical, we are directly training for our final task, i.e., plant instance segmentation. We pre-train all the networks on $6,747$ images from PhenoBench labeled by our approach. The images are taken from partially herbicided fields, i.e., there are both crops and weeds from different days with respect to the images in the annotated dataset. We then fine-tune on PhenoBench using 100%, 50%, 25%, 10%, and 5% of the training set and report the results on the validation set. We compare our results to those of the network without pre-training and initialized with MS COCO pre-training.

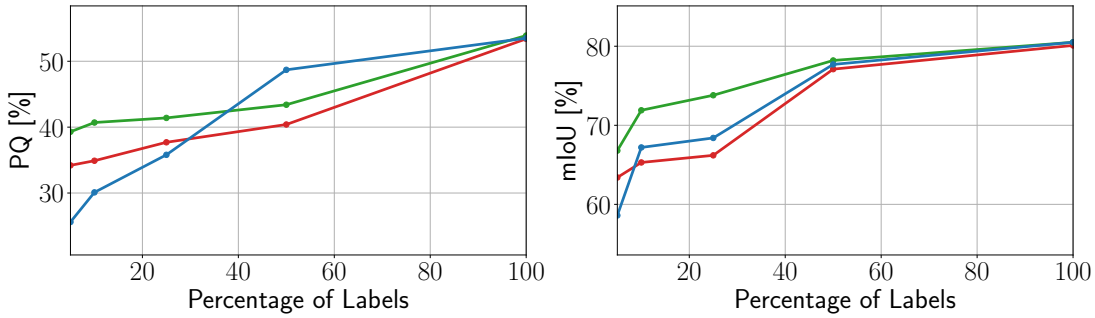
In Fig. 6.9, we can see that when using only 5% of the training data, our pre-training weakens the performance of Mask R-CNN. This is mainly because our labels do not provide the correct semantic information, thus needing more weed



(a) pre-training results using Mask R-CNN



(b) pre-training results using PanopticDeepLab



(c) pre-training results using HAPT

Figure 6.9: Results after fine-tuning the three backbones without any pre-training, initialized with our pre-training, and with the pre-training on MS COCO.

Table 6.4: Results on the validation sets of GrowliFlower and PhenoBench, both independently and together, for HAPT. We train on PhenoBench only and with additional labels provided by our approach on different datasets.

Extra Data	Test Set		IoU [%]			PQ [%]
	PB	GF	soil	crop	weeds	
none	✓		**99.2**	**90.5**	**50.4**	**53.4**
		✓	84.0	0.0	-	0.0
	✓	✓	94.1	60.3	**50.4**	35.6
Sugar Beets [4]	✓		98.7	86.4	36.4	48.3
		✓	86.9	45.6	-	28.6
	✓	✓	94.6	72.8	36.4	41.7
Corn [5]	✓		98.4	89.5	38.8	48.9
		✓	84.0	24.7	-	21.8
	✓	✓	93.6	67.9	38.8	38.9
GF (Train) [106]	✓		98.8	89.0	27.2	45.2
		✓	**94.0**	**78.5**	-	**39.8**
	✓	✓	**97.2**	**85.5**	27.2	**43.4**

labels to correct the weights of the semantic decoder. We achieve similar results for the supervised MS COCO pre-training, which diminishes the performance until 50% of the real data is available. In this case, the problem is most likely the domain gap between the pre-training and the final application. If we use 10% or more of the training data, our pre-training boosts the performance over all investigated scenarios. In all the investigated scenarios, using the weights from the MS COCO pre-training has lower performance compared to our approach. This makes us believe that the domain gap has a bigger impact than the size of the dataset, since MS COCO has 118K images, while we used only 7K images.

6.2.4 Ablation Studies

In our experimental evaluation, we identified recurring patterns and observed results that we wanted to analyze further to gain a deeper understanding of the strengths and weaknesses of both our proposed methods. These include factors related to the generalization capability of the graph-based segmentation approach and the influence of its hyperparameters. Moreover, we consider it essential to investigate the limitations of our semantic annotations, especially in the presence of weeds, to validate the applicability of our methods in real-world scenarios. In the following, we present a series of experiments targeted to explore these aspects

Table 6.5: Results of the vegetation IoU and PQ of the approaches tuned on SugarBeets and tested on PhenoBench. We highlight in bold the best results for each metric. All results are given in %.

	Approach	IoU [%]	PQ [%]	#Params	#Hyperparams
heuristic	graph-based [62]	3.7	0.0	-	3
	HUE [81]	70.8	0.0	-	2
	ExG [228]	72.1	20.4	-	3
	Our approach	**80.1**	**43.2**	-	7
learning	MR [83]	24.1	7.8	43.9M	2
	PD-S [39]	76.3	11.5	7.7M	7
	Weyler [222]	68.4	32.4	2.25M	4
	HAPT [183]	75.7	17.5	2.4M	7

in detail.

6.2.4.1 Generalization of Graph-Based Segmentation

In this experiment, we illustrate that the graph-based segmentation method, enhanced with our modifications and post-processing, can generalize well even if the hyperparameters are fine-tuned on a different dataset. Generalization is a common problem for neural networks [119, 197, 236] that has been often addressed using domain adaptation techniques [66, 232] to close the domain gap between the training and testing datasets. Most of these techniques require retraining the network on labeled data from the target domain to incorporate the new information. It is usually harder to overfit the underlying data distribution with heuristics-based approaches since they use a smaller number of hyperparameters. For this reason, they are generally more robust to changing conditions. When restricting ourselves to the plant instance segmentation task, the environmental conditions that cause adaptation problems are the illumination, the presence of shadows, the growth stage, soil color, and texture. Our results suggest that heuristics-based approaches, if domain-specific, can achieve good results on new and unseen fields without re-tuning hyperparameters.

In Tab. 6.5, we show the results of the approaches when their parameters are optimized on the SugarBeets dataset and tested on the PhenoBench dataset. We also include the results of supervised deep-learning approaches trained on the SugarBeets dataset. This allows us to show how much supervised approaches suffer from a shift in the domain, even when using the same crop species. Our approach is more robust to the domain shift and obtains the best performance on both metrics, even compared to learning-based approaches. We are shifting

from a dataset with no weeds labeled, no overlapping leaves, almost no shadows, and plants of a homogeneous growth stage to a dataset with weeds and crops of different sizes, and in the presence of overlaps and shadows.

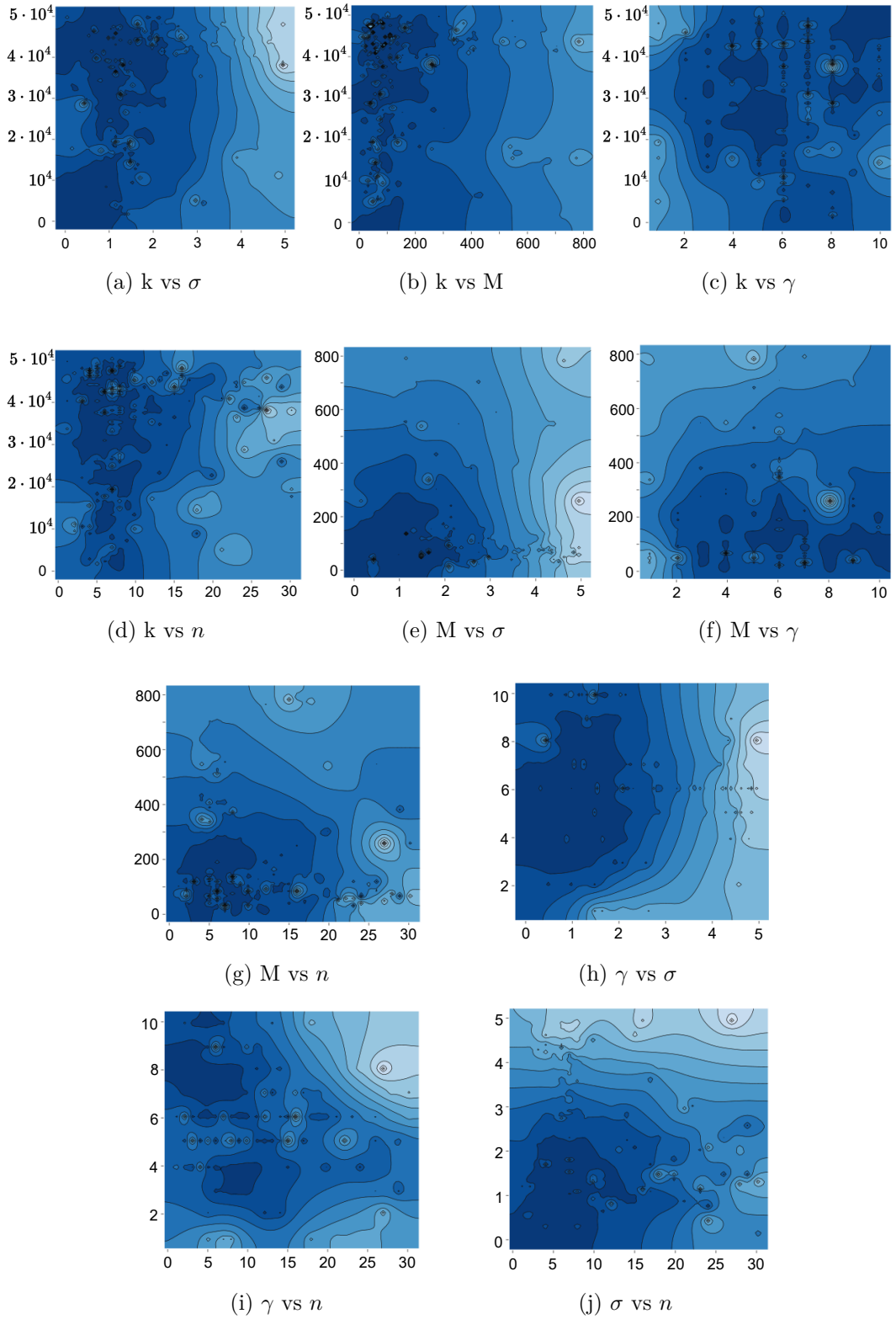
If we compare these results to those in Tab. 6.1, we can see the difference in performance given by fine-tuning the hyperparameters on a different dataset. The difference in performance is given by the domain gap and how much the approaches overfit on the original data used for the optimization. As expected, all methods suffer from a drop in performance. The approach by Felzenszwalb [62] fails, obtaining an IoU of 3.7% and a PQ of 0%. Similarly, the HUE [81] segments the vegetation, but it fails to distinguish instances with a PQ of 0%. Among the heuristic approaches, our method is the most robust, retaining the ability to segment the vegetation and identify plant instances. Due to the domain gap, our method loses 4.5% of IoU and 4% of PQ. When there is no knowledge about the deployment environment or access to labeled data. Our approach can still perform the task, outperforming both the heuristics- and learning-based baselines.

6.2.4.2 Hyperparameter Influence on Graph-Based Segmentation

We run the whole pipeline on the datasets presented in Sec. 6.2.1 to evaluate the influence of the different hyperparameters. The optimizer can evaluate the importance of the parameters, changing them in the desired ranges and computing how much the changes influenced the target, i.e., the PQ. For this experiment, we keep the thresholds of the excess green τ_{ExG} and the aspect ratio τ_a fixed, and optimize only σ, k and M, the kernel size γ and number of iterations n for the dilation. The evaluator collects the PQ and the hyperparameter configuration for all runs; thus, we can plot the results for all the values in the ranges defined for the hyperparameters and can interpolate between the different runs. Since we run the experiment for 5 different hyperparameters, we visualize the plots for the 10 different pairs in Fig. 6.10, where white corresponds to low PQ and dark blue to high PQ.

As expected, some hyperparameters allow achieving high PQ for large range of values, i.e., for each possible value of k there are values of the other hyperparameters that yield high PQ, while other hyperparameters require to be in a specific range, i.e., M should be smaller than 200, and n should be smaller than 15. By analyzing the plots, we confirm that the values we used in our experiments are in the dark blue ranges, thus yielding high PQ. Keeping the excess green τ_{ExG} and the aspect ratio τ_a fixed, we can estimate more plausible ranges for the other hyperparameters and avoid areas yielding low performance. We also notice that the hyperparameters fixed in Sec. 6.1.4 always give the possibility to tune the other hyperparameters to get the best possible result.

In Fig. 6.11, we visualize the importance of the 5 hyperparameters to opti-



(a) k vs σ (b) k vs M (c) k vs γ

(d) k vs n (e) M vs σ (f) M vs γ

(g) M vs n (h) γ vs σ

(i) γ vs n (j) σ vs n

Figure 6.10: Evaluation of the different hyperparameters combinations, where a darker color correspond to an higher panoptic quality on the images of the training set. The small black stars correspond to real trials for which the PQ has been computed, the other values are computed by interpolating the results from the trials.

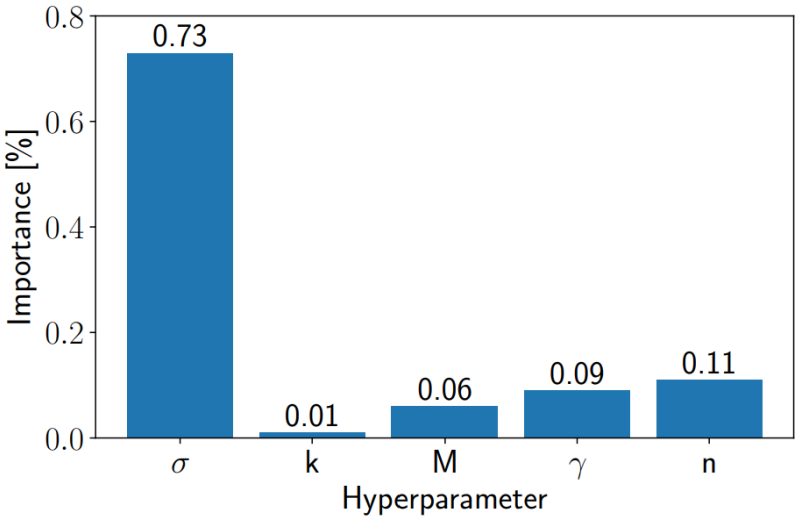


Figure 6.11: Estimated importance of the hyperparameters of the graph-based pipeline over 100 trials over the three datasets.

mize, estimated by Optuna [7] during the optimization. It turns out that σ is the most important hyperparameter, which can be explained by the fact that it defines the amount of smoothing of the image, thus making the pixel-level instance segmentation less precise as σ grows.

6.2.4.3 Filtered Semantic Supervision

Our methods target plant instance segmentation without considering the semantic class of the instances, i.e, we do not predict if a vegetation component is a weed or a crop. If we want to use our generated labels and also train for semantic segmentation, the results are sub-optimal when the network is not presented with enough weeds.

To better investigate how to address this problem, we ran all experiments again with Panoptic DeepLab without computing the semantic loss when the label comes from our approach. In this way, we remove the systematic error we would otherwise introduce. The results of this ablation study are shown in Tab. 6.6. The results indicate that when there is a sufficient amount of real labels, the network predicts better semantic masks. However, when there are only a few real labels available, this disables any supervision of the semantic head, leading to worse performance. We do not report the results with zero labels because if the network does not have access to semantic labels, neither from our generated labels nor from the manually annotated dataset, it cannot learn semantics at all.

We investigate the impact of semantics on our pre-training by conducting a similar experiment. In this experiment, we only consider the vegetation IoU and PQ, i.e., treating weeds and crops as one unique class. This brings our pre-

Table 6.6: Results on the PhenoBench dataset, using PD-S with different percentages of real labels. For each percentage of labels, weshow the results computing the semantic loss over all labels or only over the manually annotated labels. We report in bold the best results per metric for each label percentage.

Percentage of	Semantic Loss on		IoU [%]			PQ [%]
Real Labels	Real Labels	Generated Labels	soil	crop	weed	
75	✓	✓	97.8	79.7	18.1	23.1
	✓		**99.2**	**86.8**	**32.6**	**29.2**
66	✓	✓	98.9	80.3	14.1	22.8
	✓		**99.2**	**85.1**	**31.2**	**28.3**
50	✓	✓	98.9	81.2	17.9	21.4
	✓		**99.2**	**84.7**	**29.0**	**25.8**
25	✓	✓	98.3	**78.2**	**4.8**	**18.4**
	✓		**98.5**	76.5	4.4	7.0

training task to be the same as the final task, without any semantic difference in the vegetation class.

We report in Tab. 6.7 the average difference in mIoU and PQ over the three approaches for the different amounts of labels used during the fine-tuning, compared to the same training evaluating for all three semantic classes. Removing the weed class always improves performance, both because the network does not need to separate the vegetation into two classes and because semantic segmentation with only two classes is easier. The biggest gap occurs in the scarcity of labels, as the network struggles learning the weed class, lowering both metrics. When comparing the results to the networks without pre-training, we note a similar trend. The results in Tab. 6.8 show that for a higher percentage of labels, the difference becomes less. The best improvement is obtained with 10% of labels, probably because with less data, it is difficult for the networks to recover from possible mistakes introduced by the generated labels during pre-training.

6.3 Discussion

Instance segmentation, specifically plant instance segmentation, has been addressed with both heuristic-based and deep learning methods. The former leverage domain-specific assumptions and do not require large annotated datasets, but their performance deteriorates when hand-tuned thresholds and rules cannot capture the high variance introduced by overlapping foliage, lighting variation, and

Table 6.7: Difference on mIoU and PQ when counting both crops and weeds as a unique semantic class (vegetation) while fine-tuning with respect to fine-tuning with the three semantic classes. The results are averaged on the three architectures.

	Percentage of Labels				
	5%	10%	25%	50%	100%
mIoU [%]	**+ 19.8**	+ 18.5	+ 18.4	+ 17.4	+ 16.5
PQ [%]	**+ 4.4**	+ 2.8	+ 3.2	+ 3.5	+ 0.8

Table 6.8: Difference on mIoU and PQ when counting both crops and weeds as a unique semantic class (vegetation) while fine-tuning with respect to the results obtained on the same task without our pre-training. The results are averaged on the three architectures.

	Percentage of Labels				
	5%	10%	25%	50%	100%
mIoU [%]	+ 4.0	**+ 6.4**	+ 1.9	+ 1.1	+ 0.5
PQ [%]	+ 4.8	**+ 6.5**	+ 5.6	+ 5.1	+ 1.0

irregular plant geometry. The latter require extensive labeled data to achieve satisfactory performance and usually do not adapt well to unseen crop types, growth stages, or field conditions. Recent vision-language models, trained on large general-purpose datasets, show promise in addressing instance segmentation in different domains. However, they also need access to labeled data when the task complexity or domain distance increases.

To loosen the reliance on annotated datasets, we propose a fully unsupervised pipeline comprising vision-language foundation models or graph-based segmentation and domain-specific post-processing. We refine the detected instances by leveraging knowledge about the plant morphology to improve the final detection quality, showing how to overcome the limitations of both zero-shot vision-language models and purely heuristics-based graph-segmentation approaches. We evaluate multiple settings in which we employ our generated plant instances to boost the performance of supervised networks while reducing the reliance on manual annotations. We also demonstrate how our labels enhance the generalization capabilities of networks, introducing additional training data to cover more diverse crop types, growth stages, and field conditions.

Our experimental evaluation suggests that our pipeline generates plant instance labels for training networks in the absence of manual labels. Our approach performs comparably to state-of-the-art fully supervised deep learning methods

without using labels. When labels are available, we recommend using our method to enhance performance and increase the generalization capabilities of the model. The best strategy depends on the amount of available labels and the testing conditions. Our results suggest that if we have very few labeled training examples from the same crop species and field, pre-training with our generated labels and then fine-tuning on manual labels yields the best performance, see Sec. 6.2.3.4. When testing conditions are unknown, training on one dataset with manual labels integrating different fields labeled with our approach is the most effective strategy, as we demonstrated in Sec. 6.2.3.3.

We note that most of the vision-language models are trained on general-purpose datasets, which is most likely also the root cause of the shortcomings we address with out refinement step. The performance of such models usually deteriorates in very complex scenarios and domains as narrow as the agricultural one, where they need to address domain-specific challenges such as dense vegetation, overlapping canopies, and morphological differences between crops and weeds. To fully leverage these models for agricultural purposes, the best option would be to explore a domain-specific foundation model. This would enhance the performance of the first step of our proposed pipeline and the overall quality of the plant instance segmentation.

An example of failure due to unreliable prediction from the VLMs is illustrated in Fig. 6.12. The left image depicts the prediction of a VLM, where a soil mask is also assigned to part of the plants and does not capture the whole soil. This prevents our approach from accurately segmenting the remaining pixels, as the vegetation and soil colors are not reliable due to the incorrect detections by the VLM. One possible solution would be to use both the VLM and the graph-based segmentation to produce two soil-vegetation masks and check where they overlap. This can help reject severe failure cases and improve the overall results.

The right side of Fig. 6.12 represents a limitation of our approach, which splits the instances based on the expected ratio of the plant bounding box. Since the plant is not fully visible in the image the resulting instance does not match our expected aspect ratio. The aspect threshold τ_a is in this case a limiting factor, since it assumes that the plant is always fully visible in the image. We performed all experiments with the same threshold $\tau_a = 1$, however, different crop species or data acquisition procedures may require adapting the threshold to capture the new expected shape of the crops. Finally, our approach struggles to detect weed instances when they are connected to plants because they are smaller and can get eroded in our refinement step.

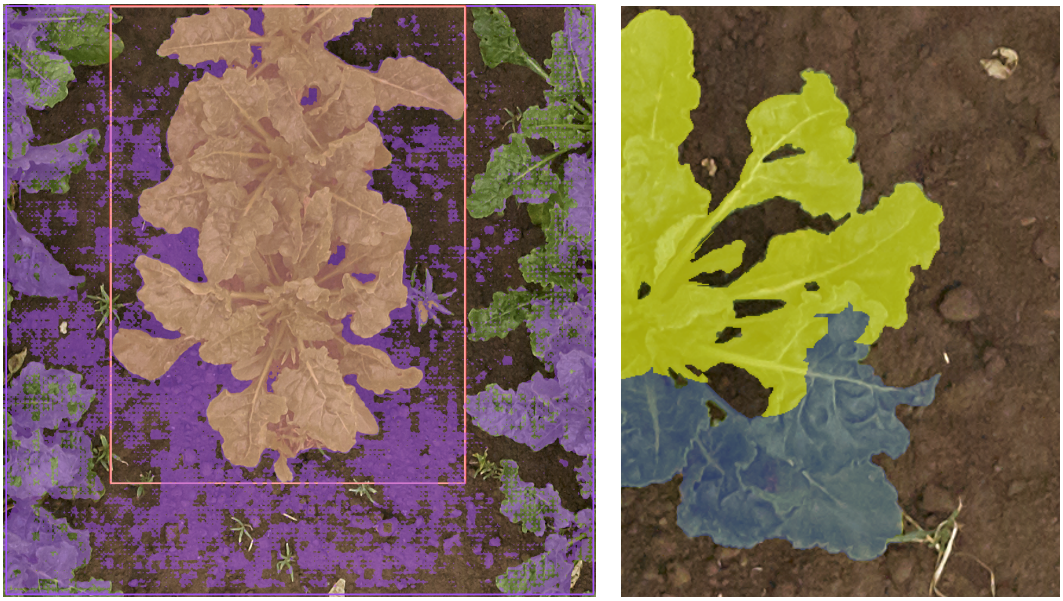


Figure 6.12: Two failure cases of our approach. On the left, the VLM fails in correctly detecting the soil, which is also assigned to most of the plants. Starting from this prediction, it is almost impossible to get a satisfactory plant instance segmentation. On the right, the instance is not respecting the expected aspect ratio because the plant is not fully visible, thus our approach splits it into two instances.

6.4 Conclusion

In this chapter, we presented an effective approach for plant instance segmentation of RGB images. Our method builds upon existing methods, such as vision-language and graph-based image segmentation methods, combined with domain-specific knowledge to improve the plant instance segmentation results without requiring additional annotated data. This integration enables us to employ common models and approaches without additional labeled data, producing high-quality plant instances through domain-guided refinement. Our pipeline enables supervision of learning-based approaches using unlabeled data as an initial training step, enhancing the performance of deep learning systems without requiring additional manual labels. We evaluated our approach on multiple datasets and provided comparisons to other existing techniques, both heuristic- and neural network-based. The experimental evaluation demonstrates that our proposed methods are competitive with current state-of-the-art approaches, and in many cases, generalize better to unseen crop fields. This makes them well-suited for application in agricultural settings when access to manually labeled data is burdensome and limited. Moreover, they can serve as a valuable pre-training stage, allowing instance segmentation networks to achieve strong performance even when fine-tuned on small, annotated datasets. Looking ahead, the next logical step in

phenotyping is moving from plant-level to leaf-level analysis, and ultimately to the estimation of leaf traits. However, since many phenotypic traits are difficult to estimate reliably in 2D due to occlusions and projection ambiguity, in the next chapter, we advance our pipeline toward 3D methods to enable more accurate and comprehensive trait extraction.

Chapter 7

Exploiting Plant Morphology for 3D Leaf Instance Segmentation

EXPANDING on the work presented in earlier chapters, we transition from 2D phenotyping approaches to a 3D perspective, where spatial structure plays a crucial role. All approaches presented in this thesis targeted the image domain, which has inherent limitations when dealing with occlusions, leaf overlap, or the complexity of plant structure. In this chapter, to better exploit the spatial information and geometrical cues of the agricultural domain, we tackle the task of leaf instance segmentation within 3D data such as point clouds [126]. This is particularly valuable in applications where plants are often only partially visible because of occlusions due to the plants' self-structure or adjacent plants. Leaf instance segmentation is the natural step after plant instance segmentation, discussed in Chapter 6. This task is crucial for estimating leaf count, monitoring the growth stages [221], and assessing the plant health conditions [10]. Accurate leaf segmentation is crucial for robotic manipulation tasks such as pruning or selective harvesting, and for phenotyping.

The phenotyping task has been addressed in the 3D scenario using supervised networks on point clouds from 3D LiDAR sensors or multi-view images. Most of these approaches build on general-purpose instance segmentation methods [100] [216]. Thus, the initial struggle to gather plant traits is still present to build a training dataset, and they do not leverage any prior knowledge about plant and leaf morphology. To fully automate the phenotyping process, the robot requires a robust perception system to acquire phenotypic traits in an automated and repeatable fashion.

Building on the key findings of the previous chapters, we propose a self-supervised pre-training method to reduce the amount of labeled data needed to achieve state-of-the-art performance on leaf instance segmentation. Supervised pre-training on point clouds [231] is still behind compared to the image-based

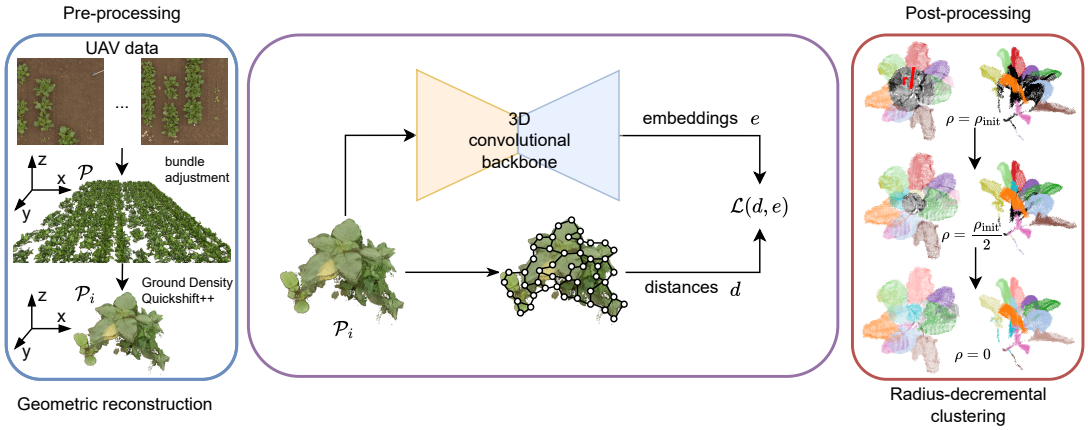


Figure 7.1: Overview of our pipeline. In the pre-processing (left), we build the point cloud $\mathcal{P}$ from UAV images using bundle adjustment and segment $\mathcal{P}$ using the method by Nelson et al. [159] into single plant point clouds $\mathcal{P}_i$. These are the inputs of our network, which learn representations computing the loss $\mathcal{L}$ on per-point embeddings $\mathbf{e}$. In the post-processing (right), we exploit domain-specific knowledge to cluster the embeddings and distinguish each leaf, starting from outer points and progressively assigning points closer to the center.

scenario, where it is common practice to pre-train networks on general-purpose datasets like ImageNet [52] or MS COCO [127]. The reason behind this gap is the challenge of annotating 3D data due to its complexity and the lack of standardized tools. This further complicates an already time-consuming and difficult process, limiting the availability of large, high-quality annotated 3D datasets. As for images, it is possible to pre-train in a self-supervised fashion also for 3D data. Self-supervised pre-training in 3D remains uncommon and, most of the time, application-specific [203]. The self-supervised pre-training we propose in this chapter is not only domain-specific, but we also make it task-specific, aligning the pre-training objective with the leaf instance segmentation task.

The main contribution of this chapter is a 3D self-supervised pre-training to differentiate each leaf of each plant. We design domain-specific augmentations and leverage task and domain knowledge to build a more specific self-supervised loss. We fine-tune on labeled data to show the improvement achieved thanks to our pre-training. We also propose a novel automatic post-processing of the self-supervised output, taking into consideration the difficulty of differentiating individual leaves – especially in the stem region – and reducing the impact of this problem on the final performance. Our task-specific pre-training enhances the performance of leaf instance segmentation and reduces the amount of labeling required. We investigate the importance of the distance information, the number of points, and the use of a second view as in common contrastive learning. We

also demonstrate that increasing the embedding size boosts the performance of our pre-training more than that of the randomly initialized network. Lastly, we evaluate our novel automatic domain-specific post-processing compared to common state-of-the-art methods.

7.1 Our Approach for Leaf Instance Segmentation Pre-Training

We propose a new unsupervised approach to pre-train a deep neural network for leaf instance segmentation in 3D point clouds. The network is part of the pipeline shown in Fig. 7.1. The pre-processing computes a point cloud from UAV images of the field and then extracts single plants leveraging the approach explained in Sec. 7.1.1. We not only make our pre-training domain- and task-specific, but we also apply the agriculture-specific augmentations explained in Sec. 7.1.2 to the single point clouds before feeding them into the backbone. In the following sections, we refer to the augmented point clouds as views and prove that our approach is effective using either one or two views. The backbone takes as input sparse tensors representing the point clouds. Each tensor consists of N points, for which we use 6 features: the first 3 represent the point position and the second 3 its color. The backbone outputs per-point embeddings, which we can use to compute the unsupervised loss $\mathcal{L}$ explained in Sec. 7.1.3, to perform a fully unsupervised leaf instance segmentation, or as features to be refined by fine-tuning using labeled data. In the latter case, we load the pre-trained weights to initialize the backbone. It is usually followed by other layers to compute the final predictions used to obtain the instances.

7.1.1 Pre-processing

We use RGB images of sugar beets collected by a UAV plus GPS information to build a dense point cloud $\mathcal{P}$ of the field via bundle adjustment [207]. The approach estimates 3D point locations and camera orientations that minimize the total reprojection error, then combines them into a unique point cloud of the entire field, with colors computed by projecting the color information from the corresponding image pixel.

To separate the point cloud $\mathcal{P}$ into individual plants $\mathcal{P}_i$, we use Ground Density Quickshift++ by Nelson et al. [159], preserving the color. The algorithm first uses Quickshift++ [101] to initialize the clusters based on the x and y coordinates and then refines these results considering the z-component. This two-step approach ensures that we do not separate stem points due to their different heights,

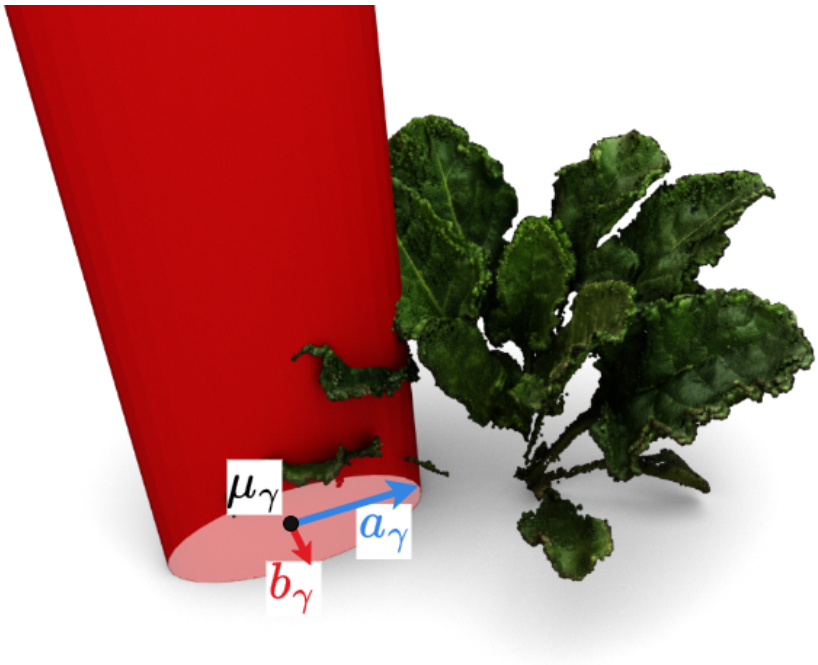


Figure 7.2: Results of the occlusion augmentation proposed in Sec. 4.1.2. All points whose x and y coordinates fall into the generated ellipse with center $\boldsymbol{\mu}_\gamma$ and axes a_γ and b_γ are removed.

and that leaves in the same area of the xy-plane are separated. We refer to the original paper [159] for more details.

7.1.2 Augmentations

Our pipeline augments the individual input point clouds $\mathcal{P}_i$ via different transformations. This is crucial in unsupervised learning, since it helps the network to focus on relevant features. We use 3D versions of common 2D augmentations – rotation, translation, adding noise, and erasing points – and our own domain-specific augmentations explained in the following, which helps us to simulate leaf occlusion and distortion. These augmentations are applied during the unsupervised pre-training to obtain better weights for initializing the network. If we use these weights to perform a fully unsupervised leaf instance segmentation, we need to revert the augmentations at inference time to have access to the real point positions, which are required by our post-processing, as described in Sec. 7.1.4.

1) Leaf Occlusion: this augmentation aims to remove all the points falling into one of Γ randomly generated ellipses. Each ellipse ϵ_γ is defined as

$$\epsilon_\gamma(x, y) = \frac{(x - \mu_{\gamma,x})^2}{a_\gamma^2} + \frac{(y - \mu_{\gamma,y})^2}{b_\gamma^2} - 1, \tag{7.1}$$

where a_γ and b_γ are the two axes, and $\boldsymbol{\mu}_\gamma = (\mu_{\gamma,x}, \mu_{\gamma,y})^\top$ is the center of the

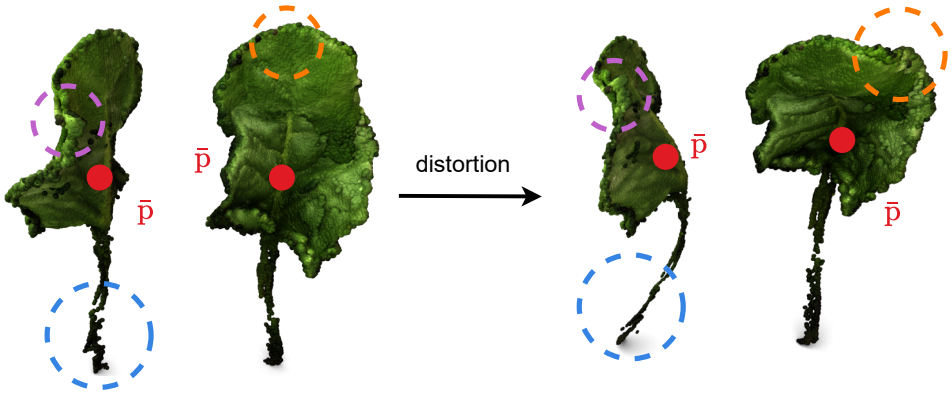


Figure 7.3: Results of the distortion augmentation proposed in Sec. 4.1.2. All points are rotated with respect to the estimated center $\bar{\mathbf{p}}$ in red. We show the side and front view for better understanding. We can see that the stem (blue dotted circle) and the leaf tip (orange) are rotated more than the areas closer to $\bar{\mathbf{p}}$ (pink).

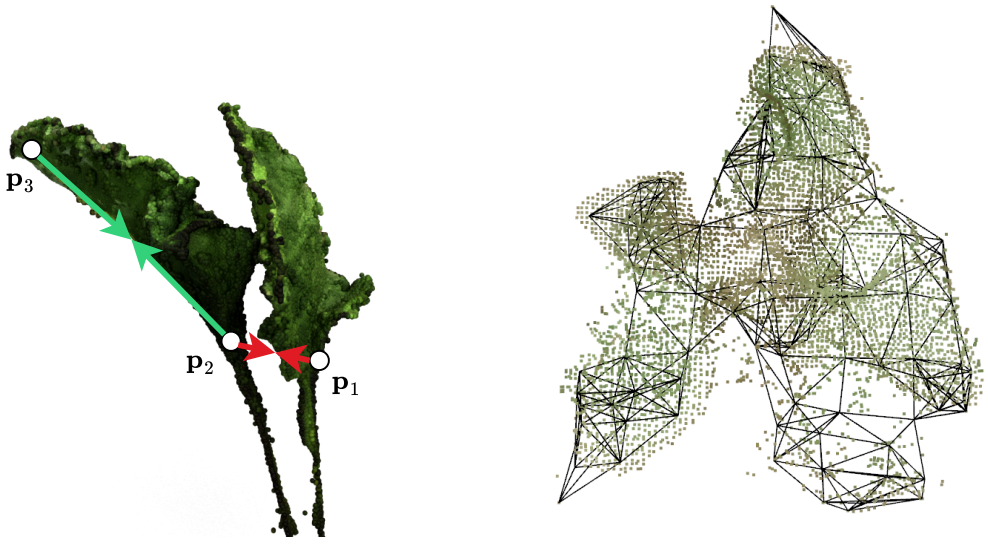
ellipse. This simulates the shape of an occlusion caused by the leaves of adjacent plants. We consider it a domain-specific variant of CutOut [55]. In our work, we use $\Gamma = 2$, since the plants are grown in rows and thus the occlusions are usually caused by plants in the same row. We randomly select either a or b as the major axis, each with equal probability. Then, we compute $\boldsymbol{\mu}_\gamma$ as the furthest point in the direction of the major axis from the center of the plant $\bar{\mathbf{p}}_i = \frac{1}{|\mathcal{P}_i|} \sum_{\mathbf{p} \in \mathcal{P}_i} \mathbf{p}$, where $\mathbf{p} \in \mathbb{R}^3$ represent the 3D positions of the points in the point cloud.

We sample a_γ and b_γ from two uniform distributions $\mathcal{U}\{0, a_{\max}\}$ and $\mathcal{U}\{0, b_{\max}\}$, where $a_{\max}$ and $b_{\max}$ are two user-defined parameters defining the maximum length and width of the leaves according to the growth stage of the plant. We project the point cloud onto the xy-plane and remove all points falling inside the ellipses. In the case of different field arrangements, we can change the parameters to have randomly spaced centers. We illustrate the result of this augmentation in Fig. 7.2, where we depict one exemplary ellipse in red.

2) Leaf Distortion: this augmentation rotates the points to imitate the movement of the leaves caused by the wind. Instead of the classical rotation of the entire point cloud, we rotate each point according to its distance to the estimated plant center $\bar{\mathbf{p}}_i$.

Given maximum rotations $\boldsymbol{\alpha}_{\max} = (\alpha_{x,\max}, \alpha_{y,\max}, \alpha_{z,\max})^\top$ about the x-, y- and z-axes, we randomly sample $\boldsymbol{\alpha} = (\alpha_x, \alpha_y, \alpha_z)^\top$, i.e., $\alpha_x, \alpha_y, \alpha_z \sim \mathcal{U}\{0, 1\}$. For each point $\mathbf{p} \in \mathcal{P}_i$, we compute the distances $d_\mathbf{p} = ||\mathbf{p} - \bar{\mathbf{p}}_i||_2$ to the plant center and define the per-point Euler angles $\boldsymbol{\alpha}_\mathbf{p} = (\alpha_{x,\mathbf{p}}, \alpha_{y,\mathbf{p}}, \alpha_{z,\mathbf{p}})^\top$ for the rotation as

$$\boldsymbol{\alpha}_\mathbf{p} = \frac{d_\mathbf{p}}{\max_{\mathbf{q} \in \mathcal{P}_i} d_\mathbf{q}} \boldsymbol{\alpha} \times \boldsymbol{\alpha}_{\max}^\top. \tag{7.2}$$



(a) (b)

Figure 7.4: In (a) $\mathbf{p}_1$ belongs to one leaf, while $\mathbf{p}_2$ and $\mathbf{p}_3$ belong to another. If we only use the euclidean distances, the embedding of $\mathbf{p}_2$ will be more similar to $\mathbf{p}_1$ than to $\mathbf{p}_3$. In (b) we show the graph built over the down-sampled point cloud using 7 nearest neighbors and $\tau_{\text{knn}} = 2\,\text{cm}$.

We can build for each point its rotation matrix

$$\mathbf{R}_{\mathbf{p}}(\alpha_{x,\mathbf{p}},\, \alpha_{y,\mathbf{p}},\, \alpha_{z,\mathbf{p}}) = \mathbf{R}_z(\alpha_{z,\mathbf{p}})\mathbf{R}_y(\alpha_{y,\mathbf{p}})\mathbf{R}_x(\alpha_{x,\mathbf{p}}), \tag{7.3}$$

where $\mathbf{R}_a(\theta)$ is the rotation matrix around axis a with angle θ. We apply the distortion as $\mathbf{R}_{\mathbf{p}}\,\mathbf{p}$, which leads to a distance-dependent rotation of points. We illustrate the result in Fig. 7.3, where different areas of the leaf undergo stronger or weaker rotations based on their distance from the leaf center, depicted in red.

7.1.3 Unsupervised Loss

We aim to learn per-point embeddings $\mathbf{e}_j^v \in \mathbb{R}^D$, with $j \in \{0, ..., N\}$, where N is the number of points in the point cloud and v is the view. In particular, we want the embeddings of the same point in different views to be identical, and points of the same leaf to be as similar as possible, facilitating their clustering.

In standard contrastive learning [37] [238], we normalize the embeddings along the feature dimension as $\widehat{\mathbf{e}}_j = \frac{\mathbf{e}_j^v}{||\mathbf{e}_j^v||_2}$ and compute the cross-correlation between each pair of points. The loss can be expressed as

$$\mathcal{L} = \sum_{r,c=0}^{N} \mathbf{\Lambda}_{r,c} - \widehat{\mathbf{e}}_r^0 \left(\widehat{\mathbf{e}}_c^1\right)^\top, \tag{7.4}$$

where $\mathbf{\Lambda} \in \mathbb{R}^{N \times N}$ is the identity matrix.

Since we aim to perform instance segmentation, we want to include spatial information and push the embeddings of close points to be similar.

To do so, we need to remove the identity matrix and use a matrix that carries the information about how similar each cosine similarity should be. This new loss function will be

$$\mathcal{L} = \sum_{r,c=0}^{N} \mathbf{S}_{r,c} - \widehat{\mathbf{e}}_r^0 \left(\widehat{\mathbf{e}}_c^1\right)^\top, \tag{7.5}$$

where $\mathbf{S} \in \mathbb{R}^{N \times N}$ specifies how similar each pair of points (r, c) should be given their distance, meaning that $\mathbf{S}_{r,c}$ is a function of the distance $\mathbf{D}_{r,c}$. This means that we can compute the loss even with just one view. Since we augment the point clouds with translations and rotations, distances must be computed from the same view to be comparable. In the following, we investigate two types of distances, Euclidean and graph distances, to include spatial information in $\mathbf{S}$.

When using the Euclidean distance between points $\mathbf{p}_r, \mathbf{p}_c \in \mathcal{P}_i$, we simply set $D_{r,c} = ||\mathbf{p}_r - \mathbf{p}_c||_2$. This makes similar the embeddings of close points, but when leaves overlap, it can happen that points of different leaves are led to have more similar embeddings than points from the same leaf, as shown in Fig. 7.4(a).

To overcome this limitation, we propose to compute the graph distance on the plant. To do so, we need to create a graph $G = (V, E)$, where V are the points and E the edges given by the k-nearest neighbor graph, i.e., each point has edges to the k closest points if their distances are smaller than a threshold τ_{knn}. We build the graph using 7 per-point neighbors and a maximum distance of $\tau_{\text{knn}} = 2\,\text{cm}$ between connected points. The result of this operation is shown in Fig. 7.4(b). We then initialize a distance matrix $\widetilde{\mathbf{D}}$ as

$$\widetilde{D}_{r,c} = \widetilde{D}_{c,r} = \begin{cases} ||\mathbf{p}_r - \mathbf{p}_c||_2 & \text{, if } (r,c) \in E \\ \infty & \text{, otherwise.} \end{cases} \tag{7.6}$$

Afterwards, we use the Floyd-Warshall [63] algorithm to traverse the graph and fill the whole matrix. Each element of row r and column c in $\mathbf{D}$ is computed from $\widetilde{\mathbf{D}}$ as

$$D_{r,c} = D_{c,r} = \min_k (\widetilde{D}_{r,k} + \widetilde{D}_{k,c}). \tag{7.7}$$

In the end, $\mathbf{D}$ has a null diagonal, positive values if two nodes are connected, and ∞ otherwise. After filling up the matrix $\mathbf{D}$, we compute the similarities as

$$S_{r,c} = \frac{1}{D_{r,c} + \xi}, \tag{7.8}$$

where ξ is an arbitrarily small value used to avoid numerical instability. We normalize the similarities as $\mathbf{S}_{r,c} = \frac{S_{r,c}}{\max(\mathbf{S})}$. Thus, all points with zero distance will

have a similarity of 1, and all other values will have a similarity inversely proportional to the distance between the points. We will evaluate both possibilities in our experiments, comparing the performance when using Eq. (7.4), Eq. (7.5) with different distances and numbers of views.

7.1.4 Post-Processing

Instance segmentation tasks can be solved by predicting embeddings to cluster, or instance centers and offsets to such centers for each point. Since it is hard to supervise the predictions of centers in the absence of labels, our pre-training produces embeddings. Thus, to evaluate the fully unsupervised approach, we need an embedding-based clustering post-processing.

In the agricultural setting, points close to the center of the plant are more complex to assign correctly since many different leaves connect to the same stem. Considering this problem, we propose a novel automatic post-processing that starts clustering the leaves from the outer points. When not specified, we cluster via agglomerative clustering [64], and use the cosine similarity "sim" to compute the similarity. Our post-processing consists of two steps:

1) Define radii to cut. We compute the center of the plant $\bar{\mathbf{p}}_i = (\bar{\mathbf{p}}_{i,x}, \bar{\mathbf{p}}_{i,y}, \bar{\mathbf{p}}_{i,z})^\top$ and the distances to the farthests points on the x and y axes as

$$d_x = \max_{\mathbf{q} \in \mathcal{P}_i} |\mathbf{q}_x - \bar{\mathbf{p}}_{i,x}| \quad \text{and} \quad d_y = \max_{\mathbf{q} \in \mathcal{P}_i} |\mathbf{q}_y - \bar{\mathbf{p}}_{i,y}|. \tag{7.9}$$

We then define the initial radius of the area to cut as

$$\rho_{\text{init}} = \frac{\min(d_x, d_y)}{2}, \tag{7.10}$$

which allows us to distinguish the tips of the leaves as a starting point.

2) Radius-decremental clustering. At each step, we decrease the size of the radius as

$$\rho = \rho_{\text{init}} - \frac{\rho_{\text{init}}}{n_{\text{steps}}}, \tag{7.11}$$

where n_{steps} is the number of steps we want to make in the post-processing operations. We cluster only the embeddings of non-clustered points with distance from $\mathbf{p}$ greater than ρ. For each of the new M clusters $\hat{\mathcal{C}}_k$ with $k \in \{1, \ldots, M\}$, we compute the maximum similarity with respect to the existing clusters $\mathcal{C}$ using the embeddings mean, i.e.,

$$\hat{s} = \max_i \ \text{sim}\left(\frac{1}{|\hat{\mathcal{C}}_k|} \sum_{m \in \hat{\mathcal{C}}_k} \mathbf{e}_m, \frac{1}{|\mathcal{C}_i|} \sum_{j \in \mathcal{C}_i} \mathbf{e}_j \right). \tag{7.12}$$

If $\hat{s}$ is higher than a threshold τ_{cluster}, we merge the clusters; otherwise, we create a new cluster. We iterate this step until $\rho = 0$ and all points are assigned.

7.2 Experimental Evaluation

The main focus of the approach presented in this chapter is a novel self-supervised task-specific approach for pre-training neural networks for 3D leaf instance segmentation. The results of our experiments confirm that our task-specific pre-training improves the performance on leaf instance segmentation and reduces the amount of labeling required. We successfully demonstrate that the distance information and number of points are more important than the use of a second view, and our approach has a greater benefit from a larger embedding size than the randomly initialized network. The experimental evaluation conducted on the different post-processing suggests that using domain knowledge also in this final step outperforms the instance segmentation obtained via common state-of-the-art methods, even when the same network's predictions are available.

7.2.1 Experimental Setup

Datasets. We recorded $3,566$ images using a UAV on a $50\,\text{m} \times 46\,\text{m}$ field. We compute the point cloud of the field via bundle adjustment. For pre-training, we use $2,616$ point clouds of plants, extracted via the pre-processing steps described in Sec. 7.1.1. For training and testing, we use the BonnBeetClouds3D dataset, introduced by Marks et al. [142].

Metrics. We evaluate the leaf instance segmentation using the mean Average Precision (mAP) [60], introduced in Chapter 4. In this case, we use 3D points instead of pixels, and we only have one semantic class, i.e. leaf, but the metric definition remains the same. Thus, we compute the overlap between sets of points belonging to predicted and ground truth instances.

Training details and hyperparameters. We use AdamW [130] with weight decay 10^{-6} and initial learning rate $2 \cdot 10^{-3}$ for 100 epochs in all experiments. We use a batch size of 48, thanks to gradient accumulation. We build the graph using 7 nearest neighbors and a maximum distance of $\tau_{\text{knn}} = 2\,\text{cm}$.

Baselines. We evaluate our approach using two instance segmentation baselines to compare network performance when trained from scratch and when pre-trained with our method. The first baseline is PointGroup by Jiang et al. [100], which predicts per-point offsets and semantic classes. The network's predictions are clustered and filtered by an auxiliary network to produce the final instances. The second baseline is the method by Marks et al. [143], which also predicts offsets but includes the prediction of a confidence score to prioritize fewer but more accurate predictions. In our experiments, we omit the confidence score to ensure a fair comparison with PointGroup on all ground truth leaves.

To evaluate the performance of our pre-training against other 3D pre-training techniques, we benchmark against two publicly available pre-training approaches.

Table 7.1: Results for the leaf instance segmentation with different pre-training approaches and embedding size $D = 3$.

Pre-training	# views	mAP [%]
none	—	36.8
point-to-point	2	41.2
Euclidean	2	41.8
graph	2	42.0
graph	1	**44.3**

The first is DepthContrast by Zhang et al. [243], which is a self-supervised contrastive pre-training exploiting point- and voxel-level features. The second is SegContrast by Nunes et al. [162], a self-supervised contrastive pre-training based on the extraction of segments using heuristics. They augment the segments to use them in the contrastive loss, learning more informative features.

7.2.2 Spatially Informed Pre-Training

The first experiment compares our different pre-training approaches and shows that a spatially informed pre-training is a better initialization for our target task. We refer to point-to-point in Tab. 7.1 as the approach following Eq. (7.4). It makes no use of spatial information and only enforces that the point has the same embedding in the two augmented views. This pre-training already outperforms the randomly initialized network. Incorporating spatial knowledge enhances the performance, confirming that our spatially informed pre-training is more aligned with the leaf instance segmentation task. Using graph distances obtains the best result, as expected considering the limitations of the Euclidean distances presented in Sec. 7.1.3. Results suggest that, when the graph is representative of the point cloud, the graph distance is a good approximation of the geodesic distance on the surface of the plant. This provides a better initialization for the network that then needs fewer iterations or labels to outperform other approaches. Since the approach with graph distances outperforms the others, we use this as the base approach to conduct all further ablations and experiments.

7.2.3 Best Use of Distances and Views

This experiment investigates whether computing the distances on the graph via the Floyd-Warshall algorithm, even if computationally demanding, improves the performance compared to using the kNN distances, i.e., $\widetilde{\mathbf{D}}$ from Eq. (7.6). On top of that, we demonstrate that the number of points used in the loss computation

Table 7.2: Results for the ablations on the graph pre-training on the number fo points used, and on the computation of the distances with the Floyd-Marshall (FW) algorithm versus the distance matrix provided by the k-nearest neighbor (kNN) algorithm.

# views	# points	mAP [%]	
		kNN	FW
2	7 000	36.3	42.0
1	7 000	36.0	40.2
1	10 000	**39.9**	**44.3**

is crucial for a good initialization, more than using a second augmented view. Tab. 7.2 shows that using one or two views does not impact the improvement gained from using the graph, as we gain approx. 4% points of mAP in both cases. Using just one view performs worse if all other parameters stay the same, but decreases the GPU usage, allowing us to use more points in the loss computation. Increasing the number of points closes the gap between the one-view and the two-views approach, outperforms the latter, and shows that using more points is more beneficial than using a two-views approach. Importantly, using two views prevents us from increasing the number of points due to the memory usage of the second view and all its embeddings. The result of the experiment suggests that using the Floyd-Warshall algorithm is worth its computational cost, and that, considering the memory usage of each point and its embedding, using only one view with more points provides a better network initialization.

7.2.4 Label Requirement Reduction

This experiment aims to show the capability of our approach to reduce the required amount of labeled data for the leaf instance segmentation task. We initialize the backbones with our pre-training and use progressively fewer labels when fine-tuning. The results in Fig. 7.5 show that we can boost the performance when using all the available data, and we obtain a similar or better mAP using 45% of the labeled data or more. We also include some qualitative images obtained after fine-tuning on 50% of the labeled data in Fig. 7.6.

7.2.5 Embedding Size Scalability

The following set of experiments investigate which embedding size is most suitable for the task and if the unsupervised pre-training consistently improves the results. Increasing the embedding size provides us with more representational power but it also increases the memory usage for each point.

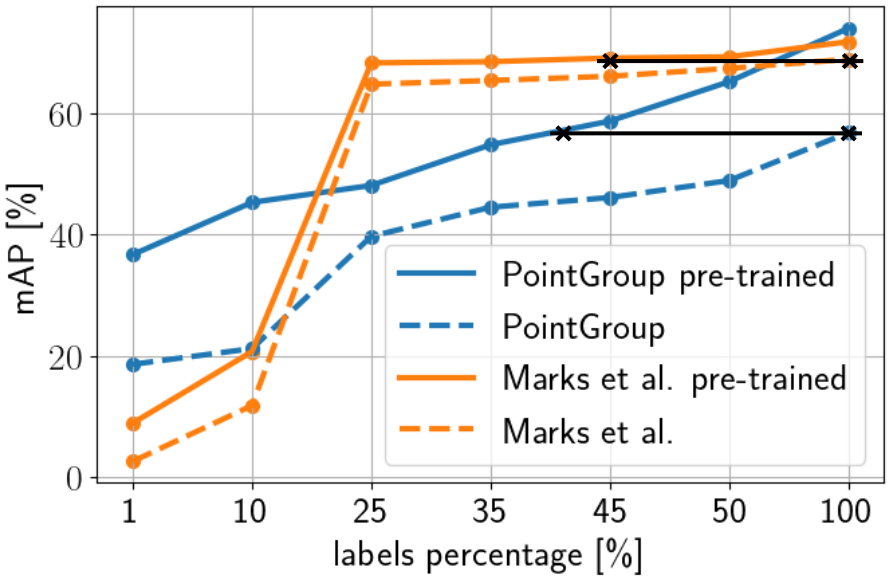


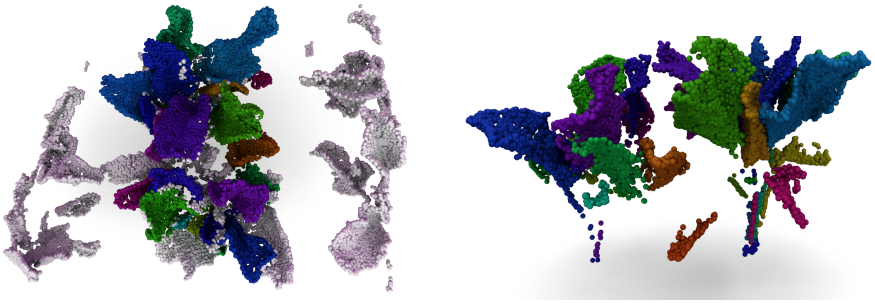
Figure 7.5: The plot of the mAP fine-tuning with different amounts of labels for the two approaches [100, 143]. In black we highlight the difference in labels from the best results without pre-training.

7.2.5.1 Number of Points or Embedding Size

Fig. 7.7 shows the results of the fine-tuning, after pre-training with the same embedding size and different number of points in the loss computation. We can see that for the first half of the plot, the more points we use, the better the performance is. However, using more than 10 000 points does not further increase the final mAP. This can also be due to the number of available points for each input point cloud $\mathcal{P}_i$, when $|\mathcal{P}_i| < 10\,000$ the loss would use fewer points, not impacting the final performance.

7.2.5.2 Increasing the Representational Power

Tab. 7.3 shows the result of the fine-tuning with different embedding sizes for our approach using graph distances, the Euclidean distance-based approach, and the randomly initialized network. We use the same number of points for embedding sizes 3 and 24 to make a fair comparison. The highest embedding size prevented us from using 10 000 points because of the higher memory usage. We are sure that if a more powerful GPU is available, using 10 000 points for the last experiment would further improve the performance. However, Tab. 7.3 shows the best results we could obtain using the same machine. We can see that using the graph distances is consistently better for all the embedding sizes. Increasing the embedding size yields higher mAP for all the approaches due to the greater representational power. However, we notice that the gap between the graph pre-training and the



(a) (b)

Figure 7.6: Results of the approach by Marks et al. [143] pre-trained with our method and fine-tuned on 50% of the labels. In (a), a top view where points with no prediction are in white. In (b), a side view removing the points without prediction.

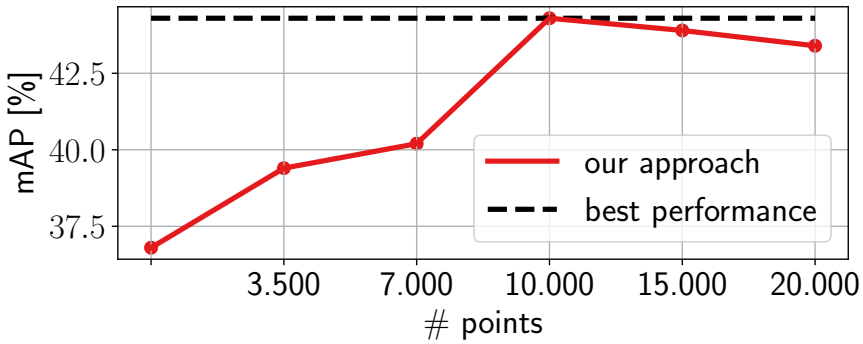


Figure 7.7: The plot of the mAP fine-tuning different pre-trainings and number of points used in the loss, with embedding size $D = 3$. The curve shows a linear dependency before saturation occurs.

other approaches is also increasing with the embedding size. This suggests that the graph pre-training can learn the plant structure from the point positions, thus providing a better initialization.

7.2.6 Domain-Specific Pre-Training vs. Representational Power

In this experiment, we aim to evaluate our pre-training against state-of-the-art methods, which are not trained on the domain-specific data but have more representational power. Specifically, DepthContrast [243] uses an embedding size of 96 with 10,000 points, and SegContrast [162] uses an embedding size of 128 with 20,000 points. Despite the higher embedding size and number of points, the second part of Tab. 7.3 shows that both methods underperform compared to our best configuration.

Table 7.3: Results for the mAP on the leaf instance segmentation with different pre-training approaches and embedding sizes D.

D	# points	mAP [%]		
		without pre-training	euclidean pre-training	graph pre-training
3	10 000	36.8	41.8	**44.3**
24	10 000	45.7	49.3	**60.2**
48	8 000	56.8	61.0	**74.2**

Approach	# points	D	mAP [%]
DepthContrast	10 000	96	65.8
SegContrast	20 000	128	67.2

7.2.7 Automatic Post-Processing

In the last set of experiments, we evaluate the embeddings we get from our pre-training without fine-tuning. Firstly, we evaluate three different post-processing algorithms on perfect embeddings (labels) and we progressively add noise, to show how the final mAP degrades. We use DBSCAN [59] and HDBSCAN [24] as baselines. We compare them with our post-processing, as explained in Sec. 7.1.4, and with a second algorithm implemented by us and based on graph cuts [105]. This variant uses the same Step 1 of our post-processing. In Step 2, we use the graph cut operation to separate one leaf from the rest of the plant. This must be repeated for each cluster, i.e., leaf, found in the first step. Step 3 merges the clusters into the final prediction. After assessing the performance of the different post-processings, we evaluate the fully unsupervised approach.

7.2.7.1 Post-Processings Evaluation

Fig. 7.8 shows the results for the four post-processings using perfect embeddings and adding noise. We conduct two experiments, adding (i) random noise over all the samples or (ii) gaussian-shaped noise with a higher magnitude near the center of the plants, according to what we discussed in Sec. 7.1.4. Both of our post-processings outperform the baselines (approximately +50% of mAP over DBSCAN and +35% over HDBSCAN), especially when adding Gaussian-shaped noise. We can see that using graph cut performs slightly better (approximately +0.6% of mAP), but the need to build the graph does not scale well for high-resolution point clouds. The results suggest that our post-processing methods are more robust to the expected noise than usual clustering algorithms.

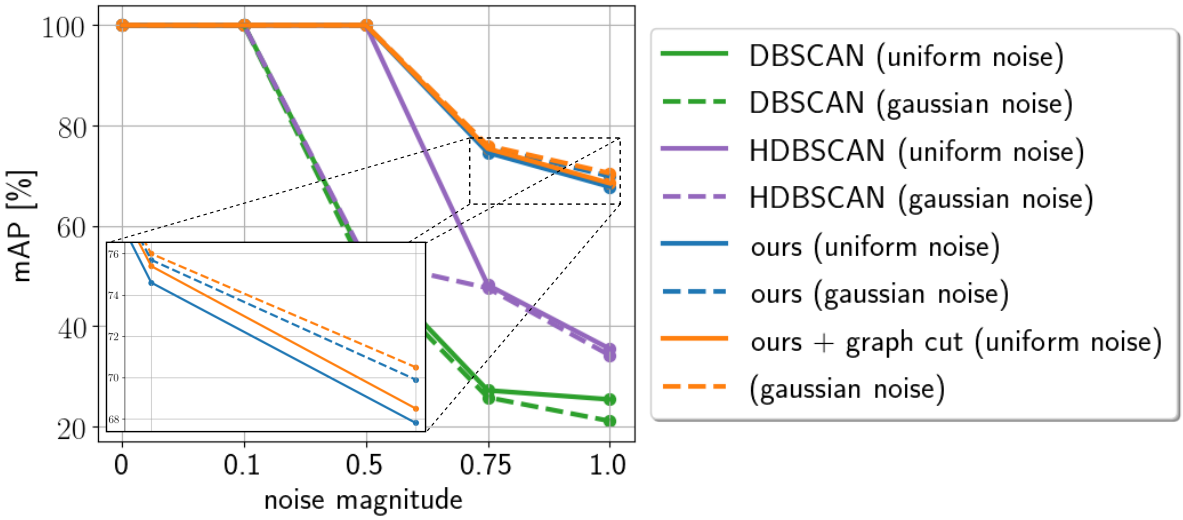


Figure 7.8: The mAP [%] of the four post-processings with increasing noise magnitude (the values stand for the maximum noise allowed) with respect to perfect embeddings.

7.2.7.2 Fully Unsupervised Embeddings Evaluation

In Tab. 7.4 we report the results for the post-processing algorithms on the embeddings predicted from our unsupervised approaches, both Euclidean and graph-based. In the first row, we provide the results obtained running HDBSCAN on the point positions, which we use as the most simple baseline. Our best performance is less than 10% worse than DBSCAN on the noisy but perfect embeddings. The results confirm that the graph approach extracts more meaningful features and that our post-processing has better performance than state-of-the-art approaches. This set the mAP for the task, without using any labels, from 2.8% to 13.6%.

7.3 Discussion

We already mentioned in Chapter 4 that the effectiveness of a pre-training depends on how aligned the data, the augmentations, and the pre-training objective are to the final domain and task. All these considerations are also valid in the 3D domain, but contrary to Chapter 4, in this chapter we aim to pre-train for a specific task; thus, we can develop a self-supervised objective well-aligned with the leaf instance segmentation task.

We target leaf instance segmentation to perform the next step in our phenotyping pipeline, which is required to extract morphological leaf traits. Our approach leverages in-domain agricultural data, domain-specific augmentations, and a self-supervised task as close as possible to the leaf instance segmentation task. In this way, we align our pre-training with our domain and with our desired perception task. We compare against state-of-the-art pre-training techniques for

Table 7.4: Evaluation of the post-processings with different embedding sizes and pre-training approaches. The first row shows the result obtained running HDBSCAN on the 3D positions only (without network).

post processing	embedding size	mAP [%]	
		Euclidean pt	graph pt
HDBSCAN	3D positions	2.8	
DBSCAN	24	5.7	6.9
HDBSCAN	24	6.9	11.3
ours	24	7.9	12.2
ours + graph cut	24	9.3	**12.4**
DBSCAN	48	6.3	10.9
HDBSCAN	48	7.5	11.3
ours	48	8.1	12.3
ours + graph cut	48	11.3	**13.6**

the entire network architecture and demonstrate that, even with a more compact representation, i.e., a smaller embedding size D, we achieve superior performance, confirming all the findings from Chapter 4. Our experiments suggest that, even if the network has access to the point positions, including spatial information in the pre-training is crucial to learn how to distinguish between different objects in the scene. In our case, this enables the use of a single view while pre-training and reduces the memory footprint of the training. We use this to increase the representational power of the embeddings and the number of points for which we compute the loss, but we could use it to increase the batch size, providing higher-quality gradients during optimization.

We believe that the best pre-training setting highly depends on the resolution of the input point clouds, the growth stage of the plants to segment, and the available computational resources. High-resolution point clouds allow the computation of the loss on more points because of their density, but close points will probably provide a similar gradient, increasing the memory footprint without having a great impact on the final result, as shown in Sec. 7.2.5.1. The plant growth stage must be taken into consideration as it directly affects the number of points and the complexity of the task, i.e., later growth stages usually exhibit more leaves and occlusions. This could guide the decision on the needed embedding size and the tuning of the hyperparameters for building the graph to differentiate overlapping leaves. On top of that, all these choices must be compatible with the available computational resources, which pose a constraint on increasing the embedding size, the number of points, and the granularity of

the graph. The results in Tab. 7.3 demonstrate that a higher embedding size is a better choice than increasing the number of points. We did not perform an extensive evaluation of the hyperparameters for building the graph because our dataset comprised plants from the same growth stage. Furthermore, even if a more fine-grained graph can better guide the pre-training, we could experience the same problem of Euclidean distances, connecting close leaves, and diminishing the quality of the learned embeddings.

The results in Tab. 7.4 are promising, as we boost the unsupervised leaf instance segmentation from 2.8% to 13.6% of mAP. However, we still require fine-tuning over a labeled dataset to obtain a reliable segmentation. A more accurate graph representation could improve the performance, even considering the aforementioned risk. Recomputing the graph during training with different hyperparameters could enhance the network capabilities and avoid the consistent use of a graph representation of unknown quality. On top of that, instance segmentation is now commonly performed by predicting centers and offsets. This is not trivial in the unsupervised setting, as we need to know which points belong to which leave to supervise centers and offsets. Changing our pre-training to such a paradigm would increase the alignment with the supervised approaches, enhancing the usability of the learned features. Nevertheless, this would highly depend on the quality of the centers and offsets used as pre-training labels.

7.4 Conclusion

In this chapter, we proposed a novel, task- and domain-specific self-supervised pre-training strategy for leaf instance segmentation, and a novel embedding-based post-processing. Unlike general-purpose pretraining approaches, our method is tailored to the structural and semantic features of the agricultural domain. Our method exploits the large amount of data that is easy to collect and tries to reduce the labeling effort required to obtain state-of-the-art performance on the leaf instance segmentation task. The approach relies on domain-specific data augmentations and a task-specific loss, plus domain-specific automatic post-processing.

We implemented and evaluated our approach, provided comparisons to other pre-training approaches and to state-of-the-art post-processing methods. The experiments suggest that our pre-training based on graph distances is a better initialization for the target task, and it boosts the final performance across all the investigated scenarios. We achieve better performance when using the same amount of data and computational power, and we can achieve the same performance using fewer resources. Notably, our automatic post-processing outperforms standard clustering algorithms, exhibiting greater robustness to real-world noise patterns such as uniform and Gaussian noise.

By enabling robust and efficient leaf-level instance segmentation with minimal supervision, our method paves the way for more scalable and accurate robotic crop monitoring. Now that we have reliable 3D reconstructions of individual leaves, we are well-positioned to move to the next stage: estimating biologically meaningful leaf traits. This will be the focus of the next chapter.

Chapter 8

Exploiting Leaf Morphology for Trait Estimation

In the previous chapter, we proposed an approach to improve the performance of 3D leaf instance segmentation, while reducing the reliance on labeled data. Now, with the knowledge of leaf instances, we can perform the final step of the phenotyping pipeline proposed in this thesis: extracting relevant morphological traits. Leaf trait estimation refers to the process of quantifying attributes relevant for the understanding of plant behavior affecting its throughput and resistance to stress and diseases [36,120,200]. Leaf traits, such as leaf area, leaf angle, and the leaf blade length and width, are crucial indicators of plant health and function [149, 198, 201].

Traditionally, these traits are manually measured by workers, making this procedure expensive, time-consuming, and difficult to scale [234]. This limitation is evident in agricultural datasets, which often provide only the average traits computed over a few manually selected leaves, reducing the granularity and accuracy of successive analyses. On top of that, this type of annotations are often not enough to evaluate approaches that aim to estimate per-plant traits and is even more problematic for training deep learning approaches, which usually require large amounts of labeled data for supervised training [140]

In this chapter, we address the challenge of the limited availability of training data by developing a generative approach to obtain leaf point clouds of given lengths and widths. We can then use our generated data to optimize approaches for the estimation of such leaf traits. Prior works [136,209,223] on trait estimation focused on leaf instance segmentation on images, treating the number of leaves as the main trait. However, images provide a limited understanding of angles and curvatures, which are needed to estimate the length and width of bending leaves. 3D point clouds can better capture the geometry of the leaves, allowing for more accurate estimation of geometric leaf traits, such as the leaf width and length.

Most of the approaches for 3D data are rule-based [50, 93, 141, 157] instead of data-driven [35,143] since the lack of data with reference traits does not allow for training of learning-based methods to estimate traits different from the number of leaves. However, all approaches still require fine-tuning on labeled data to achieve satisfactory performance in estimating any of the morphological traits.

The main contribution of this chapter is a novel approach for generating leaf point clouds with their associated leaf traits. Our work paves the way for developing and benchmarking the next generation of trait estimation techniques, previously limited by data scarcity. Unlike the traditional template leaf model, a mechanistic rule-based representation developed by expert plant scientists to capture the leaf morphology, we use a generative network trained on real-world data. As input, our network receives a point cloud representing a leaf skeleton with its traits as high-level descriptors. The network generates realistic point clouds of leaves as output. Training on real-world data encourages our approach to generate leaves similar to the real ones, without the need for additional expert knowledge for each different plant species. We generate new leaves by providing a skeleton of the desired length and width. In this chapter, we decompose the problem of trait estimation into two parts: a generative method produces leaf point clouds with their respective leaf width and length, and then, we use the generated data to optimize the parameters of a trait estimation approach. We illustrate how we decompose the problem in Fig. 8.1.

We benchmark our approach against other heuristics- and learning-based leaf generation methods, demonstrating that our generated leaf point clouds have a high probability of being drawn from the real-world leaves distribution. Then, since our approach addresses the challenge of data scarcity for trait estimation methods, we tune different off-the-shelf trait estimation approaches on our generated data and demonstrate that these result in more accurate and precise trait estimation of real-world leaves. We evaluate our results on multiple datasets of different crop species. In summary, this chapter validates that using our generated leaves to tune trait estimation approaches achieves better performance compared to using other generated or real-world leaf point clouds. Our experimental evaluation confirms that all generated leaf point clouds respect the leaf traits we condition on and have a high probability of being sampled from the real-world leaf distribution.

8.1 Problem Formulation

We formally define the problem before explaining our proposed method for generating point clouds of leaves. Leaf trait estimation is performed by a method that we express as a function

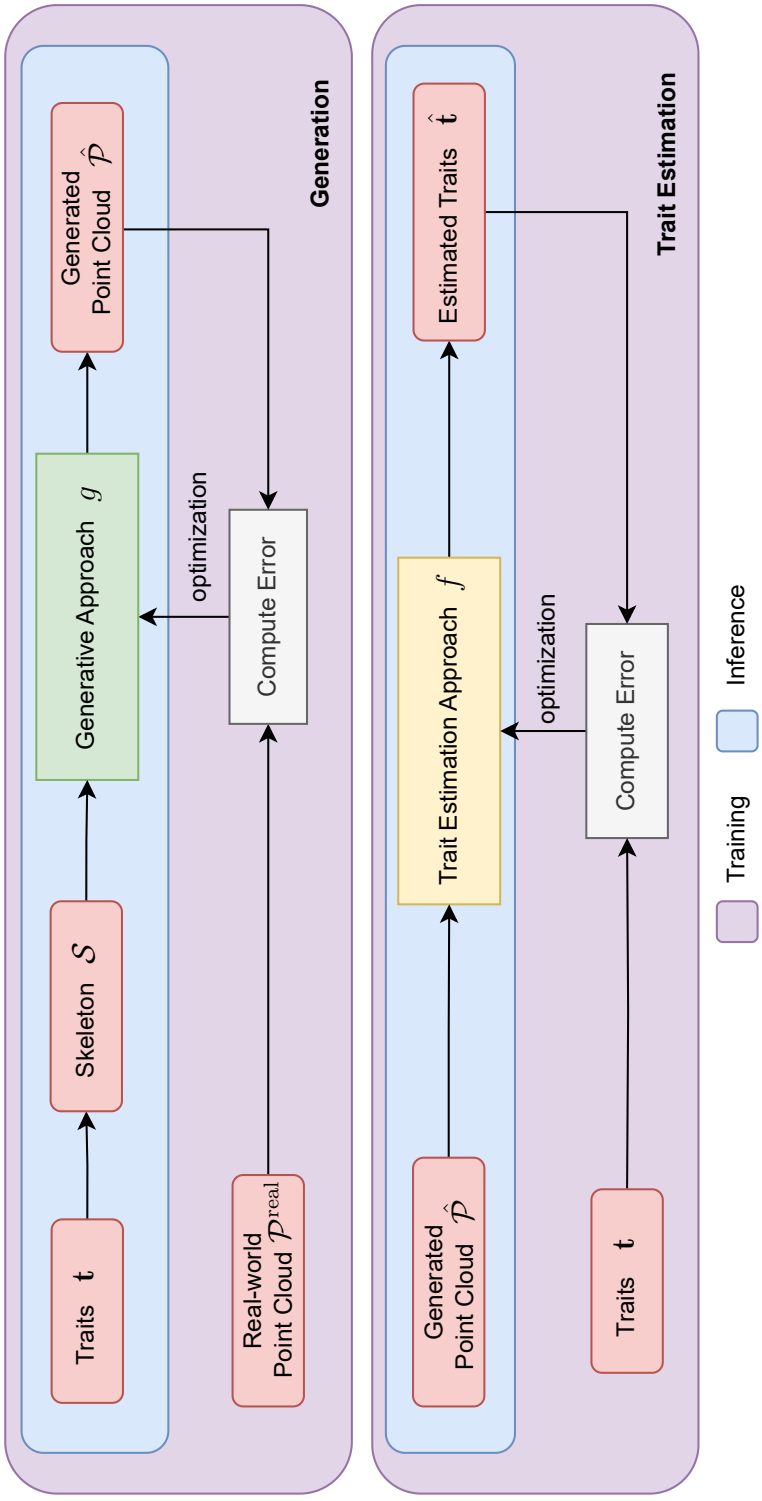


Figure 8.1: Outline of the training and inference procedure for generating a leaf point cloud $\hat{\mathcal{P}}_i$ with given traits $\mathbf{t}_i$ (top), and for estimating leaf traits $\hat{\mathbf{t}}_i$ from a given point cloud $\hat{\mathcal{P}}_i$ (bottom). We highlight the generative approach g in green and the trait estimation approach f in yellow. We color the training pipeline in purple and the inference pipeline in blue.

$$f(\mathcal{P}_i, \boldsymbol{\theta}) = \hat{\mathbf{t}}_i, \tag{8.1}$$

where $\mathcal{P}_i$ is the input leaf point cloud, $\boldsymbol{\theta}$ are the approach's parameters, $\hat{\mathbf{t}}_i \in \mathbb{R}^T$ is the vector of T estimated traits for the input leaf $\mathcal{P}_i$, each one a scalar. The f function is represented in Fig. 8.1 with a yellow block. We can find the optimal parameters of the approach given a dataset of $|\mathcal{D}|$ leaf point clouds with traits $\mathcal{D} = \{(\mathcal{P}_i, \mathbf{t}_i)\}_{i=1}^{|\mathcal{D}|}$ as

$$\boldsymbol{\theta}^*_{\mathcal{D}} = \arg\min_{\boldsymbol{\theta}\in\Theta} \sum_{(\mathcal{P}_i,\mathbf{t}_i)\in\mathcal{D}} e\left(f(\mathcal{P}_i, \boldsymbol{\theta}),\, \mathbf{t}_i\right) = \arg\min_{\boldsymbol{\theta}\in\Theta} \sum_{\mathbf{t}_i\in\mathcal{D}} e\left(\hat{\mathbf{t}}_i,\, \mathbf{t}_i\right), \tag{8.2}$$

where Θ is the set of possible parameters $\boldsymbol{\theta}$, and $e\left(\hat{\mathbf{t}}_i,\, \mathbf{t}_i\right)$ is a function computing the error between the estimated traits $\hat{\mathbf{t}}_i$ and the ground truth traits $\mathbf{t}_i$ in $\mathcal{D}$. The error function e used may depend on the estimated traits, e.g., the cosine similarity is appropriate for the angle between the leaf and the plant stem, but not for the length of the leaf blade. The error function is the white block in Fig. 8.1. As in any optimization procedure, the final performance of the trait estimation approach f depends on the dataset's completeness and reliability. As already mentioned in the introduction of this chapter, the real-world agricultural datasets $\mathcal{D}_{\text{real}}$ associate multiple leaf point clouds $\mathcal{P}_i^{\text{real}}$ with the same average traits computed over a few manually selected leaves. We want to tackle the problem of *generating a dataset* with known traits *for each* leaf point cloud. We introduce the generative problem as defining a function

$$g(\mathbf{t}_i) = \hat{\mathcal{P}}_i, \tag{8.3}$$

that generates leaf point cloud $\hat{\mathcal{P}}_i$ for given traits $\mathbf{t}_i$. The generative approach g is represented in Fig. 8.1 as a green block. In this way, we can generate a new dataset

$$\mathcal{D}_g = \{(g(\mathbf{t}_i), \mathbf{t}_i)\}_{i=1}^{|\mathcal{D}|}. \tag{8.4}$$

This new dataset $\mathcal{D}_g$, with per-leaf ground truth traits $\mathbf{t}_i$ is used to find the best parameters $\boldsymbol{\theta}^*_{\mathcal{D}_g}$ for any given trait estimation method f. The generative function $g(\mathbf{t})$ must generate realistic data to obtain a valuable dataset and, thus, parameters $\boldsymbol{\theta}^*_{\mathcal{D}_g}$ that perform well on real-world point clouds.

8.2 Our Approach for Generating Annotated Leaf Point Clouds

We propose a novel approach that generates synthetic leaf point clouds $\hat{\mathcal{P}}$ with known traits. Instead of relying on a mechanistic model, we train a 3D convolutional neural network to generate synthetic leaf point clouds of desired traits $\mathbf{t}$.

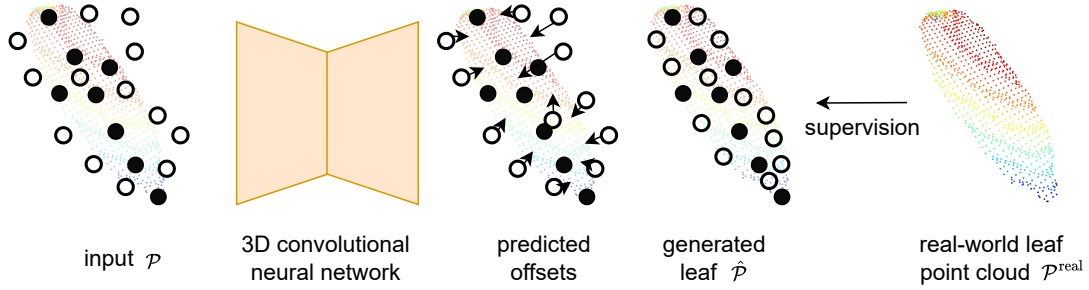


Figure 8.2: Overview of our approach. We illustrate the input point cloud $\mathcal{P}$ comprising skeleton points in black and points sampled from the Gaussian Mixture Model (GMM) in white. Our network predicts per-point offsets depicted as arrows. Adding the offsets to the point positions we obtain $\mathcal{P}$. We supervise the network using real leaf point clouds $\mathcal{P}^{\text{real}}$.

This is the g function of our problem formulation in Eq. (8.3). An overview of our approach is shown for an exemplary tomato leaf in Fig. 8.2.

In our work, we consider the leaf blade length and width as traits. Such traits are intrinsic in the leaf skeleton point cloud extracted from real-world leaves. We do not need their actual values to train our network. In Sec. 8.2.1, we illustrate how to obtain the skeleton point clouds $\mathcal{S}$ from the point clouds of real leaves $\mathcal{P}^{\text{real}}$. We do not pose constraints on how to acquire the real-world point clouds. The datasets we use have been created by means of a laser scanning system with sub-millimeter accuracy or using photogrammetric reconstruction including bundle adjustment [207] on a set of images of the field. Then, in Sec. 8.2.2, we describe how we add more points to the skeleton point cloud $\mathcal{P}^{\text{skel}}$ to capture the shape of the whole leaf and obtain the input $\mathcal{P}$ for our network. We then explain the network's architecture. In Sec. 8.2.3, we explain the loss we minimize during training, and Sec. 8.2.4 describes how we build skeletons and compute their traits from the functions that define the skeleton. Our network starts from these skeletons to generate new leaf point clouds of known leaf blade length and width.

8.2.1 Extraction of Leaves Skeletons

The first step of our approach extracts skeleton point clouds $\mathcal{P}^{\text{skel}}$ of real-world leaves $\mathcal{P}^{\text{real}}$. We use two existing approaches by Marks et al. [141] and by Magistri et al. [138] showing that our approach is independent from the skeleton extraction method. The skeleton serves as a structural backbone of the leaf, capturing the petiole, the main axis along the leaf length, and the lateral axis along the leaf width. In the literature, there is no universal definition of leaf width. Marks et al. [141] define the leaf width on their template, while for the approach by Magistri et al. [138], we define it as the width at the midsection of the leaf.

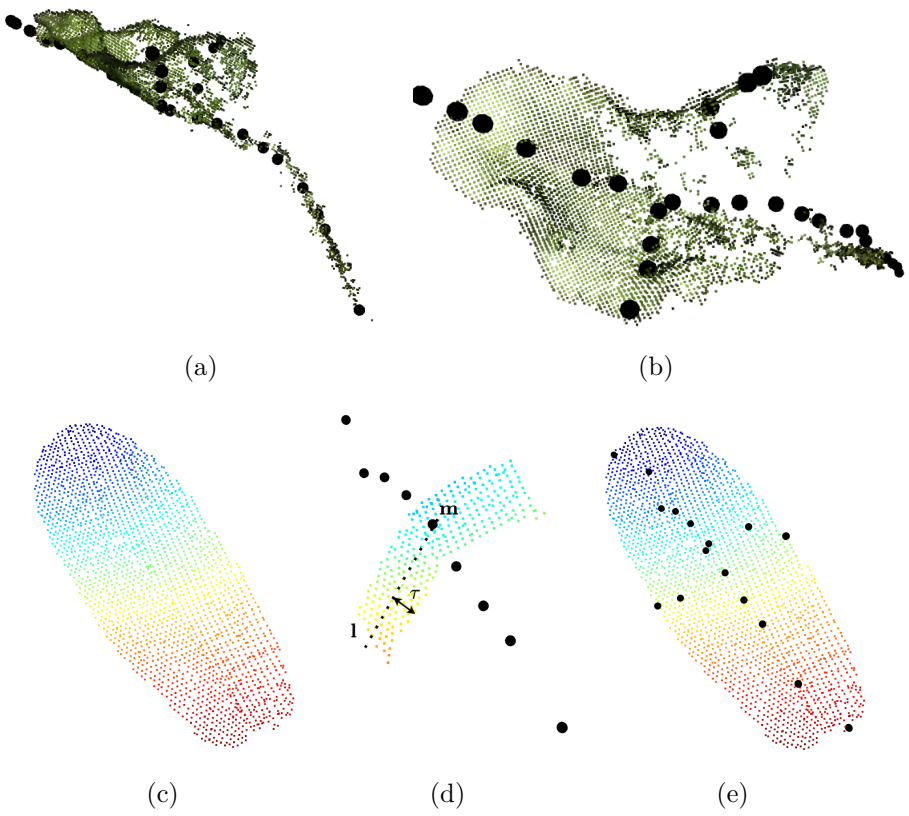


Figure 8.3: We show the extracted skeleton using the approach by Marks et al. [141] (top) for a sugar beet leaf, and the approach by Magistri et al. [138] (bottom) with our adaptation for a maize leaf. The skeleton $\mathcal{P}^{\text{skel}}$ is always shown in black circles. We show the view from the side (a) and the top (b). For the maize leaf in (c), we show the skeleton extracted along the main axis and the points of the leaf slice cut around **m** in (d). In (e), the final skeleton with main and lateral axes.

Marks et al. [141] manually define all the points and faces for a template mesh of a leaf that they deform to fit it to real leaf point clouds $\mathcal{P}^{\text{real}}$. They also define which points in the template represent the center, tip, right and left corners of the leaf, and which subsets of points represent the main axis, the lateral axis, and the petiole. The template targets sugar beet plants, and new template meshes are needed for each new crop species. Since they define the points in the template belonging to the main and lateral axis, after fitting the template mesh to the leaf point cloud $\mathcal{P}^{\text{real}}$, we use the positions of such points as points for our skeleton point cloud $\mathcal{P}^{\text{skel}}$. The top row of Fig. 8.3 shows the extracted skeleton for one exemplary sugar beet leaf.

We use the approach by Magistri et al. [138] to extract the skeleton point clouds $\mathcal{P}^{\text{skel}}$ of leaves of tomato and maize plants. The main limitation of their approach is that it only provides the points of the skeleton along the main axis of the leaf, which usually represents the leaf length. Thus, their approach does

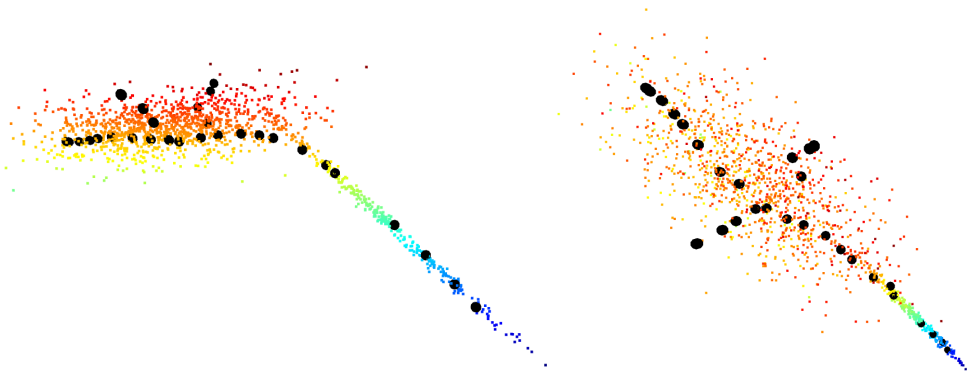
not detect the points of the lateral axis, i.e., along the width direction of the leaf. Magistri et al. [138] generate a chain of 3D points and fit it to the leaf point cloud. In the bottom row of Fig. 8.3, we show how to use their approach to also compute the points of the skeleton along the width direction of the leaf. After computing the points of $\mathcal{P}^{\text{skel}}$ along the main axis, we compute the median point $\mathbf{m} \in \mathbb{R}^3$, and the direction $\mathbf{n}$ of the lateral axis of the leaf as the second principal component extracted using principal component analysis on $\mathcal{P}^{\text{real}}$. We then cut a slice of the leaf around $\mathbf{m}$, preserving all points in the direction of $\mathbf{n}$ and removing points whose distance from the line $\mathbf{l} = \mathbf{m} + c\,\mathbf{n}$, where $c \in \mathbb{R}$, is larger than τ_l. This slice represents the central section of the leaf, from which we want to extract the points representing its width. In Fig. 8.3 (d), we show the skeleton along the main axis over the points that we keep at the end of this step. We then apply the approach only on the points in the area of interest to detect the skeleton points in the direction of the leaf width. We show in Fig. 8.3 (e) the final skeleton obtained by combining the points from this two-step approach.

8.2.2 From Skeletons to Network Outputs

We generate the leaves using a neural network, specifically a 3D U-Net [186] based on KPConv [205]. At the end of the previous section, we obtained the skeleton point cloud $\mathcal{P}^{\text{skel}}$ with $\tilde{N}$ points representing the leaf skeleton, we call $\mathbf{P}_{\text{skeleton}} \in \mathbb{R}^{\tilde{N}\times 3}$ the matrix where each 3-dimensional row represents one point of $\mathcal{P}^{\text{skel}}$. To reconstruct a complete leaf $\mathcal{Y}$, we add extra points beyond those of the skeleton to have enough points to ensure a realistic shape. We call N the total number of points in the point cloud $\mathcal{P}$ that we use as input for the network. We set $N = \tilde{N} + \delta\tilde{N}$, where $\delta \in \mathbb{Z}^+$ is a parameter that scales the number of total points according to the number of points in the skeleton $\mathcal{P}^{\text{skel}}$. We sample the extra points $\mathbf{P}_{\text{sampled}} \in \mathbb{R}^{\delta\tilde{N}\times 3}$ from a Gaussian mixture model [179] fitted to the original skeleton points $\mathbf{P}_{\text{skeleton}}$. A Gaussian mixture model is a probability distribution of density

$$p(\mathbf{p}_{\text{sampled},u}) = \sum_{j=1}^{J} \pi_j \,\mathcal{N}\left(\mathbf{p}_{\text{sampled},u}; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right), \tag{8.5}$$

where $\mathbf{p}_{\text{sampled},u} \in \mathbb{R}^3$ is the position of the u^{th} sampled 3D point, J is the number of distributions in the mixture, π_j is the probability of selecting the j^{th} distribution, $\boldsymbol{\mu}_j \in \mathbb{R}^3$ is the mean and $\boldsymbol{\Sigma}_j \in \mathbb{R}^{3\times 3}$ is the covariance of the j^{th} distribution. When collecting the real point clouds $\mathcal{P}^{\text{real}}$, we need to know if they include the petiole. When the petiole is present, we set $J = 2$; otherwise, we set $J = 1$. We require two modes when the petiole is present, as we expect one Gaussian to capture the petiole and one to capture the leaf surface. When $J = 2$, we set



(a) (b)

Figure 8.4: We show the point cloud $\mathcal{P}$, i.e., the input of our generative function g. The points of the skeleton are shown in black circles. The other points are sampled from the Gaussian Mixture Model fitted on the skeleton. We show the view from the side (a) and the top (b).

$\pi_j = 0.5 \, \forall j$. We now call $\mathcal{P}$ the point cloud obtained by adding the points $\mathbf{P}_{\text{sampled}}$ to those present in the skeleton point cloud $\mathcal{P}^{\text{skel}}$. We show the resulting point cloud $\mathcal{P}$ in Fig. 8.4 for a sugar beet leaf, where one Gaussian is fitted to the petiole and one to the leaf blade.

The output of our 3D U-Net is an offset vector $\mathbf{o} \in \mathbb{R}^3$ for each point in the input point cloud $\mathcal{P}$. We compute the positions of each $u-$th point $\hat{\mathbf{p}}_u$ in the output point cloud $\hat{\mathcal{P}}$ as $\hat{\mathbf{p}}_u = \mathbf{p}_u + \mathbf{o}_u$.

8.2.3 Loss Functions

The objective of our generative function g in Eq. (8.3), is to generate leaf point clouds $\hat{\mathcal{P}}$ respecting the traits $\mathbf{t}$ defined by the skeletons $\mathcal{P}^{\text{skel}}$, and whose points distribution is close to the real-world one. To achieve this, we combine different loss functions in the training procedure. We divide the loss functions into two main groups. The first group consists of reconstruction loss functions defined on the real-world point cloud $\mathcal{P}_i^{\text{real}}$ from which we extract the skeleton $\mathcal{P}_i^{\text{skel}}$ and the output point cloud $\hat{\mathcal{P}}_i$. The second group consists of loss functions based on the points distribution for all $\mathcal{P}_i^{\text{real}} \in \mathcal{D}_{\text{real}}$. The first group of loss functions aims to produce a leaf point cloud $\hat{\mathcal{P}}$, which respects its skeleton $\mathcal{P}^{\text{skel}}$, looks like the original point cloud $\mathcal{P}^{\text{real}}$ from which $\mathcal{P}^{\text{skel}}$ was extracted, and has a smooth surface. The second group of loss functions forces the output point cloud $\hat{\mathcal{P}}$ to have a similar point distribution compared to the point distribution in the real leaf point clouds $\mathcal{P}^{\text{real}}$. Our approach minimizes the total loss

$$\mathcal{L} = \lambda_1 \mathcal{L}_{\text{skeleton}} + \lambda_2 \mathcal{L}_{\text{chamfer}} + \lambda_3 \mathcal{L}_{\text{edges}} + \lambda_4 \mathcal{L}_{\text{smooth}} + \lambda_5 \left(\mathcal{L}_{\text{CMMD}} + \mathcal{L}_{\text{FID}} + \mathcal{L}_{\text{PR}} \right), \tag{8.6}$$

where we weight the different loss functions using $\lambda_a, a \in \{1,2,3,4,5\}$. The reconstruction loss functions are $\mathcal{L}_{\text{skeleton}}$, $\mathcal{L}_{\text{chamfer}}$, $\mathcal{L}_{\text{edges}}$, and $\mathcal{L}_{\text{smooth}}$, while $\mathcal{L}_{\text{CMMD}}$, $\mathcal{L}_{\text{FID}}$, and $\mathcal{L}_{\text{PR}}$ are distribution loss functions.

Reconstruction Loss Functions. The reconstruction loss functions use the generated leaf point cloud $\hat{\mathcal{P}}$ and the real leaf point cloud $\mathcal{P}^{\text{real}}$ from which we extracted the skeleton $\mathcal{P}^{\text{skel}}$. Their main purpose is to encourage the network to generate a leaf respecting the traits $\mathbf{t}$ defined by $\mathcal{P}^{\text{skel}}$. The first term $\mathcal{L}_{\text{skeleton}}$ forces the network to keep the skeleton points $\mathbf{P}_{\text{skeleton}}$ in their original positions, thus preserving the desired traits $\mathbf{t}$. To keep the skeleton points fixed, we enforce that the offsets predicted for those points have all components equal to zero, resulting in the loss term

$$\mathcal{L}_{\text{skeleton}} = \sum_{i}^{\tilde{N}} \mid \mathbf{o}_i \mid \mathbb{1}[\mathbf{p}_i \in \mathcal{P}^{\text{skel}}], \tag{8.7}$$

where $\mathbb{1}[\mathbf{p}_i \in \mathcal{P}^{\text{skel}}]$ is an indicator function evaluating to 1 when the points $\mathbf{p}_i$ belongs to $\mathcal{P}^{\text{skel}}$.

The second term $\mathcal{L}_{\text{chamfer}}$ is the Chamfer distance. In the literature, this distance is used to evaluate the distance between two sets of points [139,146]. We include it in our loss to enforce that the points of the generated point cloud $\hat{\mathcal{P}}$ are as close as possible to the points of the real-world point cloud $\mathcal{P}^{\text{real}}$:

$$\mathcal{L}_{\text{chamfer}} = \sum_{\mathbf{p}^{\text{real}} \in \mathcal{P}^{\text{real}}} \min_{\hat{\mathbf{p}} \in \hat{\mathcal{P}}} \left|\left|\mathbf{p}^{\text{real}} - \hat{\mathbf{p}}\right|\right|_2, \tag{8.8}$$

where $\mathbf{p}^{\text{real}} \in \mathbb{R}^3$ is a point belonging to the real-world leaf point cloud $\mathcal{P}^{\text{real}}$. We compute the Chamfer loss in both directions, i.e., we compute the closest point in $\hat{\mathcal{P}}$ for each $\mathbf{p}^{\text{real}}$ and the closest point in $\mathcal{P}^{\text{real}}$ for each $\hat{\mathbf{p}}$.

The third term $\mathcal{L}_{\text{edges}}$ is a regularization loss to enforce that the distance between neighboring points in $\hat{\mathcal{P}}$ is close to a user-defined value $\bar{l}$. This loss enforces that points are evenly distributed in space, penalizing areas that are too sparse or too dense. We compute a k-NN graph over the output point cloud $\hat{\mathcal{P}}$ defining a maximum distance $d_{\max}$ for two points to be connected, i.e., we define an edge $e_{u,v} \in E$ of length $l_{u,v} = ||\hat{\mathbf{p}}_u - \hat{\mathbf{p}}_v||_2$ between points $\hat{\mathbf{p}}_u$ and $\hat{\mathbf{p}}_v$ if $\hat{\mathbf{p}}_v \in \text{NN}_k(\hat{\mathbf{p}}_u)$, where $\text{NN}_k(\hat{\mathbf{p}}_u)$ is the set of k neighbors with distance from $\hat{\mathbf{p}}_u$ smaller than $d_{\max}$. We then compute the loss as

$$\mathcal{L}_{\text{edges}} = \frac{1}{N \sum_u |\text{NN}_k(\hat{\mathbf{p}}_u)|} \sum_{u=1}^{N} \sum_{v \in NN_k(\hat{\mathbf{p}}_u)} ||l_{u,v} - \bar{l}||_1, \tag{8.9}$$

where $|\text{NN}_k(\hat{\mathbf{p}}_u)|$ is the cardinality of the nearest neighbors of point $\hat{\mathbf{p}}_u$, and $||l_{u,v} - \bar{l}||_1$ is the absolute distance of edge $l_{u,v}$ from $\bar{l}$.

The last term $\mathcal{L}_{\text{smooth}}$ is a regularization loss to enforce the generated leaf point cloud $\hat{\mathcal{P}}$ to have a smooth surface. This loss acts like a denoising operation and penalizes single points that are too far from their neighbors and that would lead to sharp changes of the leaf surface. We use the edges computed before via the k-NN graph to compute the Laplacian matrix $\mathbf{L} \in \mathbb{R}^{N \times N}$ as

$$\mathbf{L}_{u,v} = \begin{cases} -1 & \text{, if } u = v \\ \frac{1}{|NN_k(\hat{\mathbf{p}}_u)|} & \text{, if } \exists \mathbf{e}_{u,v} \in E \\ 0 & \text{, otherwise} \end{cases} . \tag{8.10}$$

Then, we compute the Laplacian smoothing objective as $\mathbf{Q} = \mathbf{L}\hat{\mathbf{P}}$, where $\hat{\mathbf{P}} \in \mathbb{R}^{N \times 3}$ is a matrix where each row is a point $\hat{\mathbf{p}}_u \in \hat{\mathcal{P}}$. We define the loss as

$$\mathcal{L}_{\text{smooth}} = \sum_{u=1}^{N} |\mathbf{Q}_u| , \tag{8.11}$$

where $\mathbf{Q}_u \in \mathbb{R}^3$ is the u^{th} row of $\mathbf{Q}$.

Distribution Loss Functions. The second group of loss functions enforces that the distribution of the points in our generated dataset $\mathcal{D}_{\text{ours}}$ of size $|\mathcal{D}_{\text{ours}}|$, is as close as possible to the points distribution of the real-world dataset $\mathcal{D}_{\text{real}}$ of size $|\mathcal{D}_{\text{real}}|$. We use three commonly used metrics for data generation and phrase them as losses. The first term $\mathcal{L}_{\text{CMMD}}$ is the maximum mean discrepancy of the 3D CLIP embeddings [85] given by

$$\begin{aligned} \mathcal{L}_{\text{CMMD}} = & \frac{1}{|\mathcal{D}_{\text{ours}}|(|\mathcal{D}_{\text{ours}}|-1)} \sum_{i=1}^{|\mathcal{D}_{\text{ours}}|} \sum_{j \neq i}^{|\mathcal{D}_{\text{ours}}|} \langle \mathbf{v}_{r,i}^{\text{CLIP}}, \mathbf{v}_{r,j}^{\text{CLIP}} \rangle \\ & + \frac{1}{|\mathcal{D}_{\text{real}}|(|\mathcal{D}_{\text{real}}|-1)} \sum_{i=1}^{|\mathcal{D}_{\text{real}}|} \sum_{j \neq i}^{|\mathcal{D}_{\text{real}}|} \langle \mathbf{v}_{f,i}^{\text{CLIP}}, \mathbf{v}_{f,j}^{\text{CLIP}} \rangle \\ & - \frac{2}{|\mathcal{D}_{\text{ours}}||\mathcal{D}_{\text{real}}|} \sum_{i=1}^{|\mathcal{D}_{\text{ours}}|} \sum_{j=i}^{|\mathcal{D}_{\text{real}}|} \langle \mathbf{v}_{r,i}^{\text{CLIP}}, \mathbf{v}_{f,j}^{\text{CLIP}} \rangle, \end{aligned} \tag{8.12}$$

where $\mathbf{v}_{r,i}^{\text{CLIP}}$ and $\mathbf{v}_{f,j}^{\text{CLIP}}$ are the CLIP embeddings of the i^{th} real-world point cloud and of the j^{th} generated point cloud, and $\langle \cdot, \cdot \rangle$ is the cosine similarity operation. Jayasumana et al. [97] were the first to propose the use of CLIP embeddings, initially for the evaluation of generated images. Exploiting the work by Hegde et al. [85] who provide 3D CLIP embeddings trained on point cloud-image-caption triplets, we compute the CMMD on point clouds.

The second term $\mathcal{L}_{\text{FID}}$ comes from the Fréchet inception distance. As for the previous term, we first compute embeddings for all point clouds, both the real-world $\mathcal{D}_{\text{real}}$ and the generated ones $\mathcal{D}_{\text{ours}}$. We can use the model by Hegde et al. [85]

to obtain CLIP embeddings or any other neural network to extract embeddings from the point clouds. Once we have the embeddings $\mathbf{v}_{r,i}$ for all the real-world point clouds and $\mathbf{v}_{f,j}$ for all the generated point clouds, we fit Gaussians $\mathcal{N}(\boldsymbol{\mu}_r, \boldsymbol{\sigma}_r)$ and $\mathcal{N}(\boldsymbol{\mu}_f, \boldsymbol{\sigma}_f)$ to the two embedding distribution. We compute the FID as

$$\mathcal{L}_{\text{FID}} = \|\boldsymbol{\mu}_r - \boldsymbol{\mu}_f\|_2 + \text{tr}\left(\boldsymbol{\Sigma}_r + \boldsymbol{\Sigma}_f - 2\sqrt{\boldsymbol{\Sigma}_r \boldsymbol{\Sigma}_f}\right), \tag{8.13}$$

where $\boldsymbol{\Sigma}_r$ and $\boldsymbol{\Sigma}_f$ are the covariance matrices of two distributions, and $\text{tr}(\cdot)$ is the trace operation over the matrix.

The third term $\mathcal{L}_{\text{PR}}$ comes from the precision and recall metrics. These metrics have been extended for evaluating generative approaches by Kynkäänniemi et al. [115]. In the following, we shortly explain how the metrics are computed and how we adapt them to use them as losses. As for the previous distribution loss functions, we need point cloud embeddings. We call Φ_r and Φ_f the sets of features extracted from the real and generated point clouds. For each set, we estimate a manifold in the feature space by sampling a set of points and surrounding each with a hypersphere that reaches its k^{th} nearest neighbor. We then evaluate whether an embedding $\mathbf{v}$ is inside the volume estimated from the set of features Φ as

$$b(\mathbf{v}, \Phi) = \begin{cases} 1 & \text{, if } \exists_{\mathbf{v}' \in \Phi} : \|\mathbf{v} - \mathbf{v}'\|_2 < \|\mathbf{v}' - \text{NN}_k(\mathbf{v}', \Phi)\|_2 \\ 0 & \text{, otherwise} \end{cases}, \tag{8.14}$$

where $\text{NN}_k(\mathbf{v}', \Phi)$ returns the k-th nearest embedding of $\mathbf{v}'$ from Φ. We now compute the precision Pr and recall R as

$$\text{Pr} = \frac{1}{|\boldsymbol{\Phi}_f|} \sum_{\mathbf{v}_f \in \boldsymbol{\Phi}_f} b(\mathbf{v}_f, \boldsymbol{\Phi}_r) \tag{8.15}$$

$$\text{R} = \frac{1}{|\boldsymbol{\Phi}_r|} \sum_{\mathbf{v}_r \in \boldsymbol{\Phi}_r} b(\mathbf{v}_r, \boldsymbol{\Phi}_f). \tag{8.16}$$

In contrast to the previous loss functions, which are distances, we cannot use the precision and recall as they are, since we aim to maximize them. Thus, we define the precision-recall loss as

$$\mathcal{L}_{\text{PR}} = \log_{10}\left(\frac{1}{\text{Pr} + \epsilon}\right) + \log_{10}\left(\frac{1}{\text{R} + \epsilon}\right), \tag{8.17}$$

where ϵ is a small value to ensure numerical stability. In the original paper [115], the authors noticed that the score is inaccurate when measuring the quality of a generated sample that falls into an area of the manifold where only a few real samples are present. Thus, they introduce the realism score

$$\text{realism}\,(\mathbf{v}_g, \Phi_r) = \max_{\mathbf{v}_r}\left\{\frac{||\mathbf{v}_r - NN_k(\mathbf{v}_r, \Phi_r)||_2}{||\mathbf{v}_g - \mathbf{v}_r||_2}\right\}, \tag{8.18}$$

which is used to filter out elements in such sparse areas of the manifold. The higher the minimum realism score, the more we are pruning our manifold Φ_r, thus yielding accurate and higher scores.

Since the real-world data distribution does not change during training, we can easily pre-compute the target values for all the distribution loss functions $\mathcal{L}_{\text{CMMD}}$, $\mathcal{L}_{\text{FID}}$, and $\mathcal{L}_{\text{PR}}$, i.e., $\mathbf{v}_r$, $\mathcal{N}(\boldsymbol{\mu}_r, \boldsymbol{\sigma}_r)$, and Φ_r.

8.2.4 Generating New Leaves

Our generative model g takes as input the desired leaf length and width. However, our network needs as input a point cloud $\mathcal{P}$ computed from a skeleton point cloud $\mathcal{P}^{\text{skel}}$, as explained in Sec. 8.2.1. Thus, we need to define how to build a skeleton point cloud $\mathcal{P}^{\text{skel}}$ without extracting it from a real-world leaf. As mentioned in Sec. 8.2.1, the skeleton consists of three parts: petiole, main axis, and lateral axis. We construct the skeleton in the 3D Cartesian frame, building the main axis along the x direction and the lateral axis along the y direction. The petiole is a line of function

$$z_{\text{petiole}}(x) = \alpha\, x,\; x \in [x_{\min}, 0]\,, \tag{8.19}$$

where $\alpha \sim \mathcal{U}\left(\frac{\pi}{6}, \frac{\pi}{3}\right)$ and $x_{\min} \sim \mathcal{U}\,(-1, -0.25)$. Since we want a 3D point cloud, all these points still need a y coordinate, which we fix to 0. The petiole can be removed from the generative procedure when it's known that the petiole is not present in the training data $\mathcal{D}_{\text{real}}$. We do not use the petiole for the maize leaves since the petiole is not present in the used real-world point clouds $\mathcal{P}^{\text{real}}$. The central axis is defined as a hyperbolic tangent

$$z_{\text{central}}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}},\; x \in [0, 1]\,, \tag{8.20}$$

where we clamp the hyperbolic tangent between $x = 0$, where the petiole starts, and $x = 1$. All points have $y = 0$. We then scale the axis to different sizes.

We define the point where the central axis intersects the lateral axis as $\mathbf{p}_{\text{cross}} = [\mathbf{p}_{\text{cross},x}, 0, \mathbf{p}_{\text{cross},z}]^\top$, where $\mathbf{p}_{\text{cross},x} \sim \mathcal{U}(0.25, 0.75)$ and $\mathbf{p}_{\text{cross},z}$ is given by Eq. 8.20. We use a parabolic function,

$$z_{\text{lateral}}(y) = a\, y^2 + b\, y + c, \tag{8.21}$$

to represent the lateral axis, where all points have $x = \mathbf{p}_{\text{cross},x}$. To compute the parabola coefficients a, b, and c, we need 3 points. One point is $\mathbf{p}_{\text{cross}}$, and the

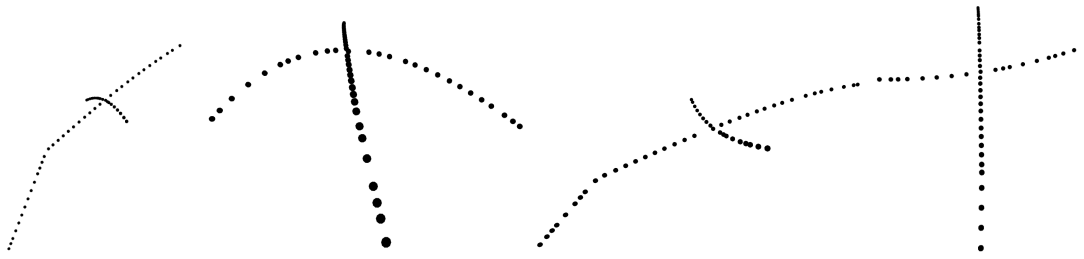


Figure 8.5: In (a) and (c) we show two generated skeletons seen from the side, while in (b) and (d) we see them from the petiole. The axis angle, the petiole length, and the extremities of the lateral axis vary.

other two are the extremes on the right and left. We define them as

$$\begin{aligned} \mathbf{p}_r &= \left[\mathbf{p}_{\text{cross},x}, 0.5, \mathbf{p}_{\text{cross},z} + z_{\text{r}}\right]^\top \\ \mathbf{p}_l &= \left[\mathbf{p}_{\text{cross},x}, -0.5, \mathbf{p}_{\text{cross},z} + z_{\text{l}}\right]^\top, \end{aligned} \tag{8.22}$$

where z_{r} and z_{l} are two distinct values sampled from $\mathcal{U}(-0.25, 0.25)$. It is important that the width of the leaf projected on the y axis is 1, we can then scale it to different sizes. The final skeleton is the collection of points sampled along the curves in Eq. (8.19), Eq. (8.20), and Eq. (8.21). We then scale this parametric skeleton by multiplying all x coordinates of the points for the length scaling factor s_l, and all the y coordinates for the width scaling factor s_w. The scaling factors are the only user-defined parameters that influence the length and width of the generated leaves. Thanks to the different randomly sampled parameters $\alpha, x_{\min}, \mathbf{p}_{\text{cross},x}, z_{\text{r}}$, and z_{l}, we obtain a large variety of leaves whose lengths and widths are centered on the user desired dimensions.

We calculate the final length and width of the leaf using the formula for computing arc lengths. The width of leaf L_{width} is computed as

$$L_{\text{width}} = \int_{-s_w}^{s_w} \sqrt{z'(y)^2 + x'(y)^2}\, dy = \int_{-s_w}^{s_w} z'(y)\, dy = \int_{-s_w}^{s_w} (2ay + b)\ dy, \tag{8.23}$$

where $x'(y) = 0$ because all points on the lateral axis have $x = \mathbf{p}_{\text{cross},x}$ and we compute $z'(y)$ deriving equation Eq. 8.21. The length of the leaf L_{length} is computed as

$$L_{\text{length}} = \int_{\hat{x}_{\min}}^{s_l} \sqrt{z'(x)^2 + y'(x)^2}\, dx = \int_{\hat{x}_{\min}}^{0} \alpha\, dx + \int_{0}^{s_l} \left(1 - \tanh^2(x)\right)\ dx, \tag{8.24}$$

where $\hat{x}_{\min}$ is the resulting minimum value for x we got multiplying $x_{\min}$ for s_l, $y'(x) = 0$ because all points of the main axis have $y = 0$, and we derived Eq. 8.19 and Eq. 8.20 to sum the length of the petiole and the length of the leaf blade. We show two exemplary skeletons in Fig. 8.5. We can see from the side view of the

skeletons in Fig. 8.5 (a) and Fig. 8.5 (c) that they have different stem lengths, leaf angles, and intersection points of the two axes. The two skeletons also present a lateral axis with opposite orientation. In the views from the petiole, Fig. 8.5 (b) and Fig. 8.5 (d), we can see that the lateral axes are skewed, not being perfectly symmetrical with respect to the main axis. One can make the skeletons more complex using different functions, or polynomials of higher grade to represent the axes. However, the results of our generative procedure suggest that our skeletons capture the characteristics of the used crop species.

8.3 Experimental Evaluation

The main focus of this work is an approach for generating 3D leaf point clouds of known length and width. Using our data improves the performance of trait estimation approaches and enables a more fine-grained analysis of crop growth and productivity. We present our experiments to demonstrate that using our generated leaves $\mathcal{D}_{\text{ours}}$ to tune trait estimation approaches outperforms using other generated leaf point clouds. All of our generated leaves respect the leaf traits we condition on and have a high probability of being sampled from the real-world leaf distribution.

8.3.1 Experimental Setup

Datasets. We use the BonnBeetClouds3D [142] dataset introduced in Chapter 7, and Pheno4D [193], a dataset captured with a laser scanning system. Both datasets provide single-leaf point clouds. We show in Fig. 8.6 (a) to Fig. 8.6 (f) images of sugar beets plants from BonnBeetClouds3D [142], in (i), (j), and from (n) to (p) tomato plants from Pheno4D [193], and in (g), (h), and from (h) to (m) we show maize plants from Pheno4D [193]. For the tomato and maize plants, we show one plant from the first date and then the same plant from the last date. We can see that leaves at the early stages are more similar, but they grow into very different shapes and structures in the later growth stages. For the sugar beets, we do not have access to earlier growth stages, so we present two distinct plants from the dataset. However, sugar beets are dicotyledonous plants, thus resembling tomato plants in the early growth stages, as the ones depicted in (g) and (h). Since BonnBeetClouds3D [142] was captured in the field and not in a controlled environment, we can see that the bottom part of the plant is occluded and not entirely present in the point cloud, which increases the challenge in estimating the correct traits.

Metrics. We evaluate the estimated traits by comparing the mean and the standard deviation estimated by all approaches when trained on the different

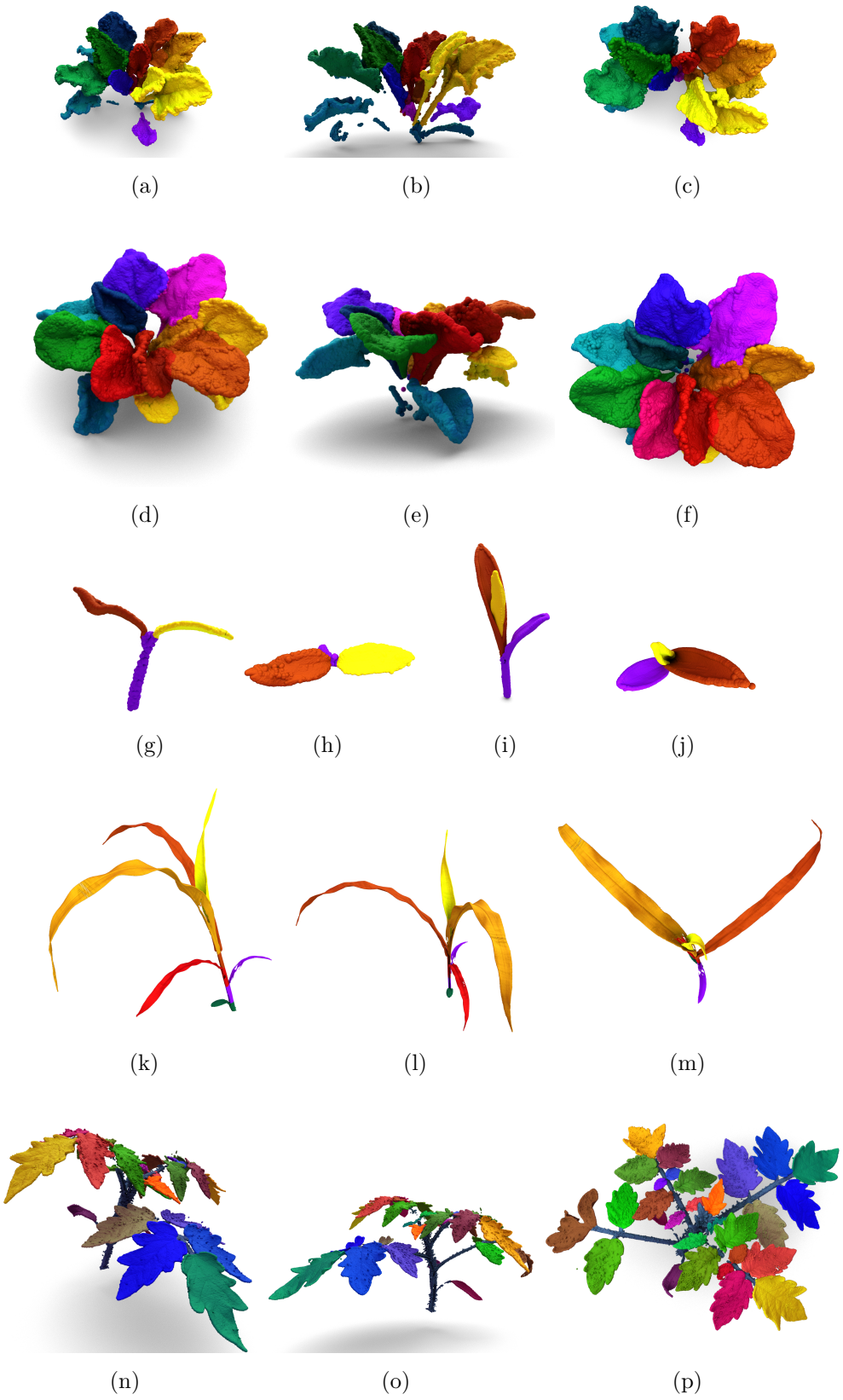


Figure 8.6: Exemplary plants of the three crop species. (a)–(f) are sugar beet plants, (g), (h), (k)–(m) are maize plants, and (i), (j), and (n)–(p) are tomato plants. For maize and tomato, we show the same plants at the first (g)–(j) and last date (k)–(p) of data acquisition. For every plant, one different color is assigned to a unique leaf.

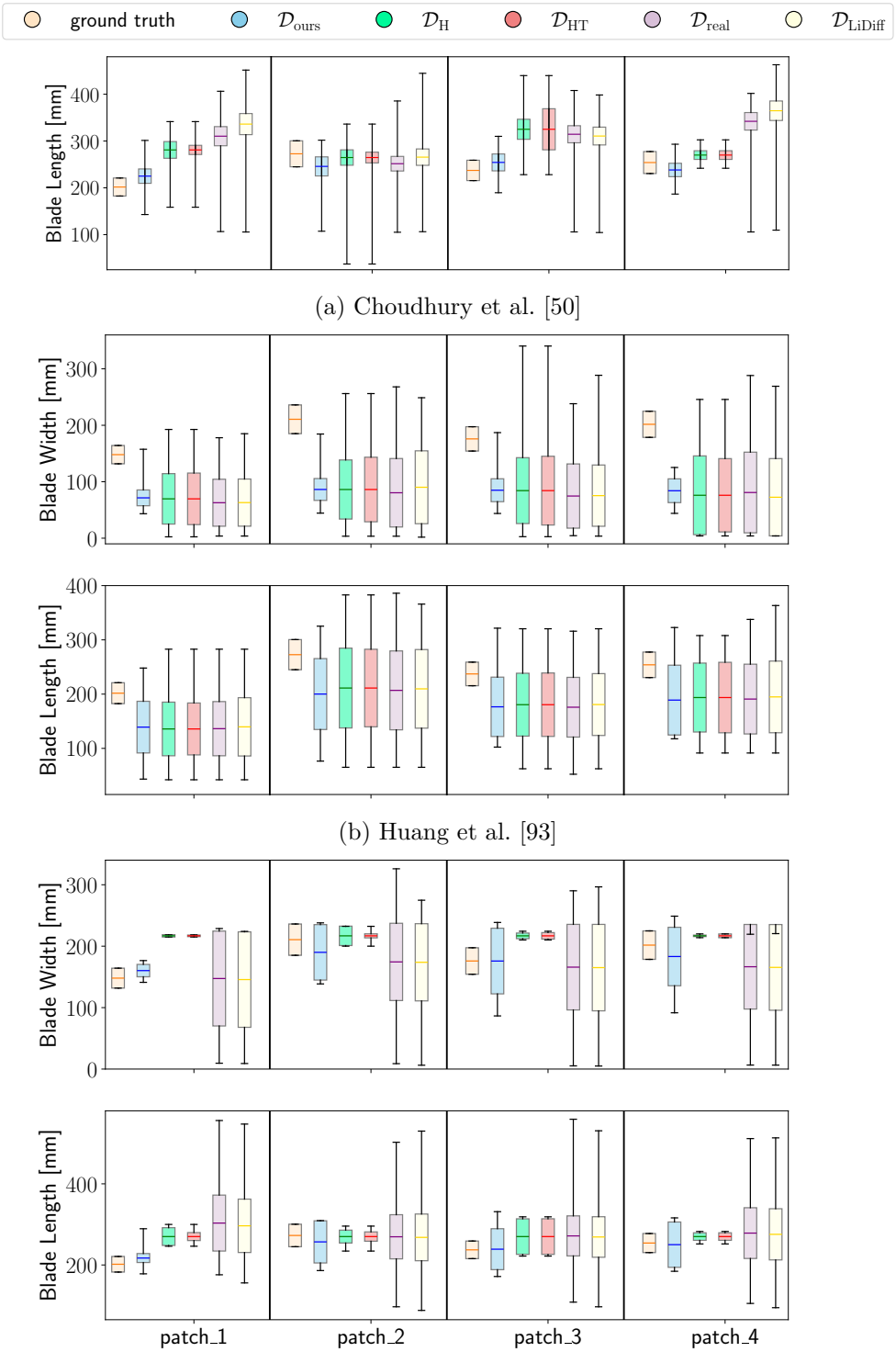
generated datasets. Additionally, we compute the Fréchet inception distance (FID) [88], the CLIP Maximum Mean Discrepancy (CMMD) [97], and the F-score computed by the precision (Pr) and recall (R) [115] explained in Sec. 8.2.3 to estimate how close the distributions of the generated and real data are. We employ the pre-trained networks from Mohammadi et al. [154] and Hedge et al. [85] to extract the embeddings $\mathbf{v}$. Both networks provide open-source code and pre-trained models. Embedding-based metrics, as the ones employed in our evaluation, are the standard approach to evaluate generative methods [19,89,115, 153,190]. These metrics provide a semantic and perceptually relevant comparison, allowing for distribution-level comparisons that would not be possible for distance metrics based on the raw points' positions. To verify that we are not generating the same leaf when conditioned on one specific skeleton, we compute the mean and standard deviation of two different metric distances between multiple leaves $\hat{\mathcal{P}}$ generated from the same skeleton input $\mathcal{P}^{\text{skel}}$.

Training details and hyperparameters. We train our network using the Adam optimizer [107] with a learning rate 0.001. In our loss, we set the weights of the different components to $\lambda_0 = 1$, $\lambda_1 = 0.1$, $\lambda_2 = 0.1$, $\lambda_3 = 10$, $\lambda_4 = 0.01$. These weights help preserve traits while producing realistic leaf point clouds. We use different scaling factors for different crop species: $s_l \sim \mathcal{U}(0.02, 0.50)$ and $s_w \sim \mathcal{U}(\frac{s_l}{4}, s_l)$ for the sugar beets, $s_l \sim \mathcal{U}(0.15, 0.90)$ and $s_w \sim \mathcal{U}(\frac{s_l}{10}, \frac{s_l}{5})$ for the maize and $s_l \sim \mathcal{U}(0.10, 0.50)$ and $s_w \sim \mathcal{U}(\frac{s_l}{2}, s_l)$ for the tomato leaves. We use $J = 2$ for sugar beets and tomato leaves, and $J = 1$ for maize leaves.

Baselines. We evaluate our approach by comparing our generated leaf point clouds $\mathcal{D}_{\text{ours}}$ to the leaves generated by three possible g functions in our problem formulation in Eq. (8.3). First, a set of leaves generated using the procedural agriculture simulation software Helios [11] exported by means of a simulated LiDAR sensor, from now on called $\mathcal{D}_{\text{H}}$. Second, we apply transformations specific to the agricultural domain from our previous work [182] to the leaves obtained from Helios to obtain a larger variety of leaves, denoted as $\mathcal{D}_{\text{HT}}$, where HT stands for "Helios + transforms". Third, we train LiDiff [163] to generate leaves conditioned on the skeletons using diffusion, from now on denoted as $\mathcal{D}_{\text{LiDiff}}$. Lastly, we also use the real-world per-plot ground truth data $\mathcal{D}_{\text{real}}$ to highlight the importance of per-leaf traits to improve the performance of the leaf trait estimation methods.

8.3.2 Trait Estimation

The first experiment evaluates how fine-tuning off-the-shelf trait estimation approaches on $\mathcal{D}_{\text{ours}}$ improves the performance compared to other datasets. We show that fine-tuning on $\mathcal{D}_{\text{ours}}$ provides better estimates in terms of mean and standard deviation without relying on costly manual annotations. We test the fine-tuned approaches on the validation set of BonnBeetClouds3D [142], which



(a) Choudhury et al. [50]

(b) Huang et al. [93]

(c) Myronenko et al. [157]

Figure 8.7: We show the leaf blade length and width estimated by the approaches for BonnBeetClouds3D [142] after tuning them on the different datasets. Each bar plot is centered on its mean, the size corresponds to its standard deviation, and we show the maximum and minimum estimated values.

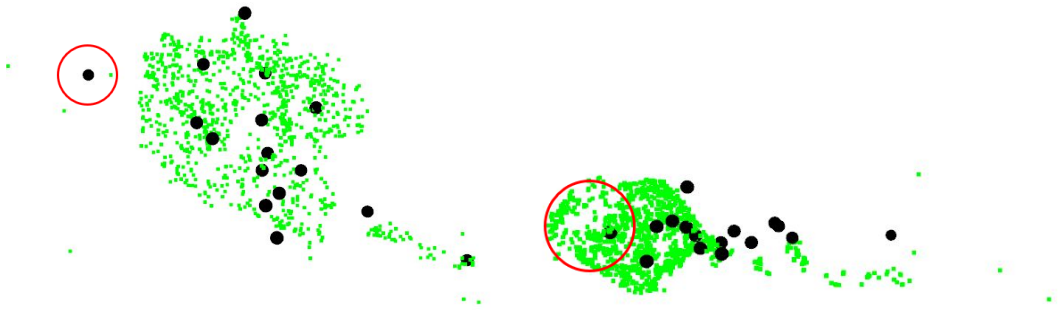
only provides mean and standard deviation per sub-areas of the field of size 1m by 1m, that they call patches.

We use three trait estimation approaches f: (i) the approach by Choudhury et al. [50] fits a polynomial to the skeleton of the leaf and then computes the leaf length via integration; (ii) the approach by Huang et al. [93] uses the principal components to define the direction of the length and width of the leaf and then computes the longest shortest geodesic distance along those directions via A* [80]; (iii) Coherent point drift [157] uses Gaussian mixture models to find the best correspondences between two set of points. Coherent point drift needs a source point cloud to deform, i.e., a leaf point cloud template, for which we use the leaf template defined by Marks et al. [141]. As explained in Sec. 8.2.1, this template mesh already defines the points belonging to the main and lateral axes, allowing us to compute the length and width of the leaf after the deformation carried out by Coherent Point Drift.

For $\mathcal{D}_{\text{H}}$, $\mathcal{D}_{\text{HT}}$, and $\mathcal{D}_{\text{ours}}$, we have per-leaf traits, while $\mathcal{D}_{\text{real}}$ only provides per-patch averages. This introduces a systematic error since all leaves from the same plot will have the same ground truth. For $\mathcal{D}_{\text{LiDiff}}$ [163], we use our skeletons of known traits, without changing the noise generation and training procedure. For our skeletons, we also know the leaf angle. However, since this was not in the ground truth measurements, we were not able to use it for evaluation purposes.

In Fig. 8.7 (a), we illustrate the results obtained by tuning Choudhury et al. [50] on the validation patches of the BonnBeetClouds3D dataset. We see that the second patch is the one where tuning over $\mathcal{D}_{\text{ours}}$ performs worse. This suggests that our generated leaf point clouds do not align well with the leaves in this patch, which is, interestingly, the one with the larger leaves. This could also explain why we are the only one underestimating the size of the fourth patch, where using $\mathcal{D}_{\text{H}}$ and $\mathcal{D}_{\text{HT}}$ results in smaller variances and better maximum and minimum estimates compared to the other patches. In general, the maximum and minimum estimates obtained by tuning on $\mathcal{D}_{\text{ours}}$ are better, even when other datasets provide a mean closer to the ground truth.

The results of Huang et al. [93] in Fig. 8.7 (b) provide similar estimates across all patches and datasets, suggesting an algorithmic limitation rather than dataset influence. The pipeline has a few hyperparameters to remove outliers and define the path cost of the A* algorithm [80] for computing the traits. We believe that the PCA-based method struggles with the complex heart shape of the sugar beet leaf and with occlusions, misidentifying the main axis and, thus, leading to estimation errors. Failures to identify the main axis would also explain the large differences in maximum and minimum estimates. The differences in the results likely depend on the dataset's resolution and sparsity, which would impact outlier detection and the computation of distances.



(a) (b)

Figure 8.8: Two leaves generated by LiDiff, where we show in black dots the skeleton used for the conditioning and in green the generated point clouds. The point clouds have the appearance of leaves, but do not respect the skeleton traits. We highlight the errors in red circles.

We show in Fig. 8.7 (c) that tuning Coherent point drift [157] on $\mathcal{D}_{\text{ours}}$ provides means closer to the ground truth but with larger standard deviations. The second patch remains problematic, confirming the trend observed for the approach by Choudhury et al. [50]. While $\mathcal{D}_{\text{H}}$ and $\mathcal{D}_{\text{HT}}$ perform better on the second patch, the uniformity of their results suggests a potential overfitting or a failure to capture the data diversity. Tuning the approaches on $\mathcal{D}_{\text{real}}$ yields diverse results but large standard deviations, especially for the blade width, likely because of the lack of per-leaf ground truths. Similarly, also using $\mathcal{D}_{\text{LiDiff}}$ shows high standard deviations, likely because the generation procedure does not preserve the traits accurately. We provide qualitative examples of the leaves generated by LiDiff in Fig. 8.8. The leaves appear realistic but do not accurately follow the skeletons, depicted as black dots, representing our desired input traits. In Fig. 8.8 (a), the leaf is shorter than expected, leaving one skeleton point outside of the leaf blade, thus providing an incorrect leaf length. In Fig. 8.8 (b), we see the opposite problem; the last skeleton point is actually inside the leaf blade that overshoots the expected leaf length. This problem is hard to address because it is not systemic, i.e., we do not always have shorter or always longer leaves. Using inaccurate traits during the optimization leads to the same problem of using the ground truth "per plot" measures.

The results show that using accurate per-leaf traits, even when artificially generated, improves the estimation results on real-world leaves. Our approach enables precise trait estimation without manual labeling or expensive expert knowledge. This is crucial for breeders and agronomists assessing plant traits linked to crop growth and productivity. However, ambiguity in defining leaf width (e.g., midsection vs. widest point) complicates evaluation. Mismatches in width

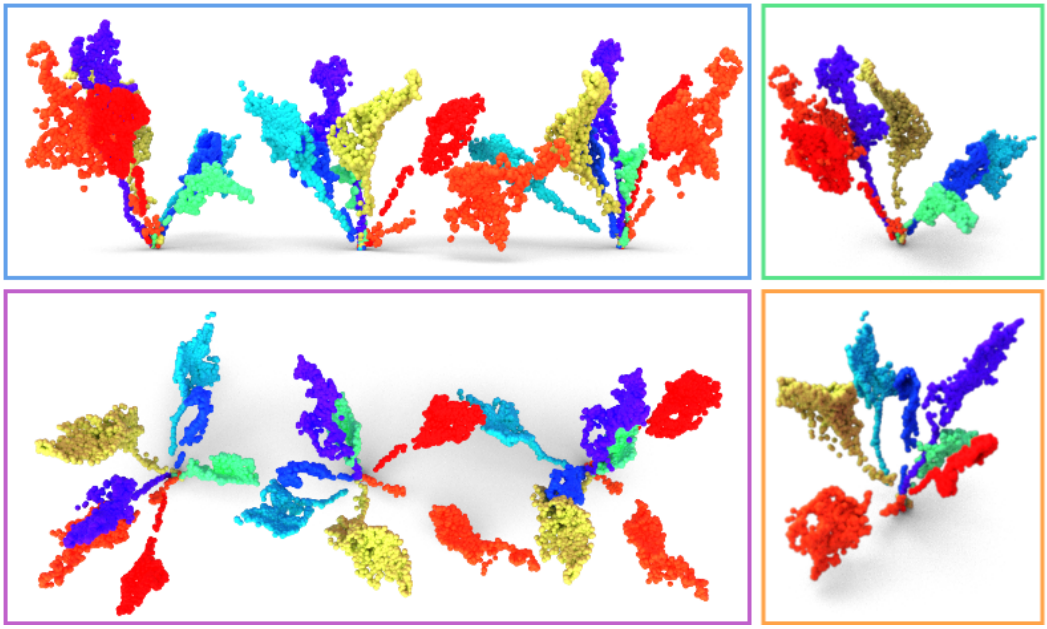


Figure 8.9: Examples of sugar beet leaves generated by our approach with different leaf angles, stem lengths, and blade lengths and widths. We show a side (blue rectangle) and a top view (purple rectangle) of three plants generated using the same leaves with different orientations and positions. We show zoomed in views of the first (green rectangle) and second plant (orange rectangle).

definitions across datasets, generative models, and estimation methods introduce systematic errors, highlighting the need for standardization in trait measurement.

8.3.3 Realistic Data Generation

The second set of experiments assesses how closely our generated leaf point clouds match real-world distributions. We demonstrate that our approach generates leaf point clouds with features similar to real-world data, making them valuable for tuning trait estimation approaches to use in real-world scenarios. Additionally, our method generated diverse leaves while maintaining specified blade length and width, bridging the gap between simulated and real-world data. As detailed in Sec. 8.3.1, we evaluate the generated point clouds with the metrics explained in Sec. 8.2.3. They compare distributions of embeddings, which we extract using the two different pre-trained networks mentioned in Sec. 8.3.1.

8.3.3.1 Leaf Distribution

We use the validation set from BonnBeetClouds3D, from now on called SugarBeets dataset, and the unlabeled plants from Pheno4D as real-world target point clouds. We illustrate examples of the leaves generated by our approach trained on BonnBeetClouds3D in Fig. 8.9, where we use the information from the skele-

Table 8.1: Evaluation of the Fréchet inception distance (FID), F-score, and CLIP Maximum Mean Discrepancy (CMMD) for the leaf point clouds generated by the different approaches compared to the test sets. Our approach outperforms the others on most metrics across the different datasets.

Dataset	Generated Data	FID ↓	F-score ↑	F-score + CLIP ↑	CMMD ↓
SugarBeets	$\mathcal{D}_{\text{H}}$	312.84	0.01	0.01	169.71
	$\mathcal{D}_{\text{HT}}$	14.01	**0.36**	0.01	28.45
	$\mathcal{D}_{\text{LiDiff}}$	26.89	0.35	0.11	22.05
	$\mathcal{D}_{\text{ours,rec}}$	108.73	0.03	0.17	20.46
	$\mathcal{D}_{\text{ours}}$	**13.71**	0.21	**0.20**	**19.53**
Maize	$\mathcal{D}_{\text{H}}$	11.28	0.19	0.02	29.39
	$\mathcal{D}_{\text{HT}}$	0.17	0.39	0.17	16.36
	$\mathcal{D}_{\text{LiDiff}}$	1.05	0.22	0.09	65.14
	$\mathcal{D}_{\text{ours}}$	**0.12**	**0.55**	**0.36**	**11.51**
Tomato	$\mathcal{D}_{\text{H}}$	7.89	0.01	0.06	28.26
	$\mathcal{D}_{\text{HT}}$	5.29	0.02	0.06	24.16
	$\mathcal{D}_{\text{LiDiff}}$	6.66	0.05	0.10	65.42
	$\mathcal{D}_{\text{ours}}$	**2.76**	**0.15**	**0.17**	**12.39**

tons to merge our generated leaves in different plants. We compute all metrics for $\mathcal{D}_{\text{ours}}$, $\mathcal{D}_{\text{H}}$, $\mathcal{D}_{\text{HT}}$ and $\mathcal{D}_{\text{LiDiff}}$. Since LiDiff requires conditioning on skeletons but does not provide a skeleton generation procedure, we use the skeletons of the training set to generate new leaves, potentially giving it an advantage over methods relying on domain expertise or procedurally generated skeletons. Tab. 8.1 shows the results of the CMMD, FID, and F-score. For the F-score, we use both feature extractors, i.e., the network by Mohammadi et al. [154] and by Hedge et al. [85], to isolate network-specific influences. Fitting a Gaussian on the CLIP embeddings of the real-world point clouds was failing and starting with non-default initializations provided inconsistent results, thus we do not include the FID metric with CLIP embeddings. Since the improved precision and recall metrics depend on the number of neighbors used to construct the real and generated data manifolds (Φ_r and Φ_f), we evaluate the F-score across multiple values of k, specifically $k \in \{2, 4, 8, 16, 32, 48, 64, 96\}$. We then report the mean F-score over these values to provide a more stable and robust estimate.

We see that applying our domain-specific transforms to $\mathcal{D}_{\text{H}}$ improves all metrics across all datasets. The results are generally better on Pheno4D, likely because this dataset consists of leaves at different growth stages compared to the SugarBeets dataset, which was recorded in one day. Overfitting to the exact growth stage could lead to a boost, explaining the results of LiDiff, which uses

the skeleton of the training point clouds. Our approach outperforms the others across most of the investigated scenarios, except for FID + CLIP and the F-score on the SugarBeets dataset, where the feature extractors yield conflicting results. For the SugarBeets dataset, we also provide an ablation study on our approach, namely $\mathcal{D}_{\text{ours,rec}}$, where we keep everything the same but we train using only the reconstruction losses, without the distribution ones. We can see that using the distribution losses improves all metrics, especially the FID and the F-score, which is 7 times better. This proves the contribution of our distribution loss functions and the importance of supervising the distribution embeddings while generating the leaf point clouds.

Given the low F-score values in Tab. 8.1, we compute the realism score as in Eq. 8.18 and re-evaluate the generative approaches on the SugarBeets dataset, where the two feature extractors contradict each other. We noticed that the number of samples filtered out by the realism score was high, especially for low values of k. Tab. 8.2 shows the F-score filtering out elements with low realism and considering only results where more than half of the generated leaves were used. Most results improve by increasing the minimum realism score, but many approaches fail when the minimum accepted realism is too high. When realism exceeds 1.0, only our generated leaves consistently allow F-score computation with both models, indicating strong alignment with real-world distributions. This explains why tuning on our data enhances leaf trait estimation. The feature extractors still disagree, highlighting the need for a standardized feature extractor, as it exists in the image domain, or even a domain-specific one that could better capture important features specific to agriculture in the data.

8.3.3.2 Leaf Variety

Our method generates leaf point clouds from pre-defined skeletons. However, we want to ensure diversity by generating different leaves also when given the same skeleton. This enhances the dataset variety without altering skeleton-building procedures or training multiple generative networks g. A diverse dataset is crucial during optimization of the trait estimation approaches to avoid overfitting to common samples. In this experiment, we input the same skeleton multiple times and compare the generated leaves by computing two distances. First, we use the Chamfer distance. Second, we compute the meshes from the leaf point clouds via ball pivoting [18] and measure the surface differences as the differences of the distances from the meshes to randomly sampled 3D points. In Fig. 8.10, we show a simplified 2D example of the point-to-mesh distance.

In Tab. 8.3, we report the mean chamfer and point-to-mesh distances on 10 leaves generated with the same skeleton averaged on 10 runs. Since we cannot condition the Helios software on a skeleton, we report the results only for our

Table 8.2: F-score computed for all the approaches using different values of realism to filter out the outliers. When less than half of the samples were valid, we do not report any result (-). The more samples we filter out, the higher the metrics. Our approach is the only one that always provides enough samples in the dense area of the distribution.

Realism	Generated Data	F-score ↑	F-score + CLIP ↑
0.0	$\mathcal{D}_{\text{H}}$	0.01	0.01
	$\mathcal{D}_{\text{HT}}$	**0.36**	0.01
	$\mathcal{D}_{\text{LiDiff}}$	0.35	0.11
	$\mathcal{D}_{\text{ours}}$	0.21	**0.20**
0.5	$\mathcal{D}_{\text{H}}$	0.01	0.01
	$\mathcal{D}_{\text{HT}}$	**0.48**	0.02
	$\mathcal{D}_{\text{LiDiff}}$	0.33	0.11
	$\mathcal{D}_{\text{ours}}$	0.22	**0.23**
1.0	$\mathcal{D}_{\text{H}}$	0.02	0.02
	$\mathcal{D}_{\text{HT}}$	0.45	0.04
	$\mathcal{D}_{\text{LiDiff}}$	**0.64**	0.11
	$\mathcal{D}_{\text{ours}}$	0.29	**0.55**
1.5	$\mathcal{D}_{\text{H}}$	-	-
	$\mathcal{D}_{\text{HT}}$	**0.80**	-
	$\mathcal{D}_{\text{LiDiff}}$	-	-
	$\mathcal{D}_{\text{ours}}$	0.29	**0.90**

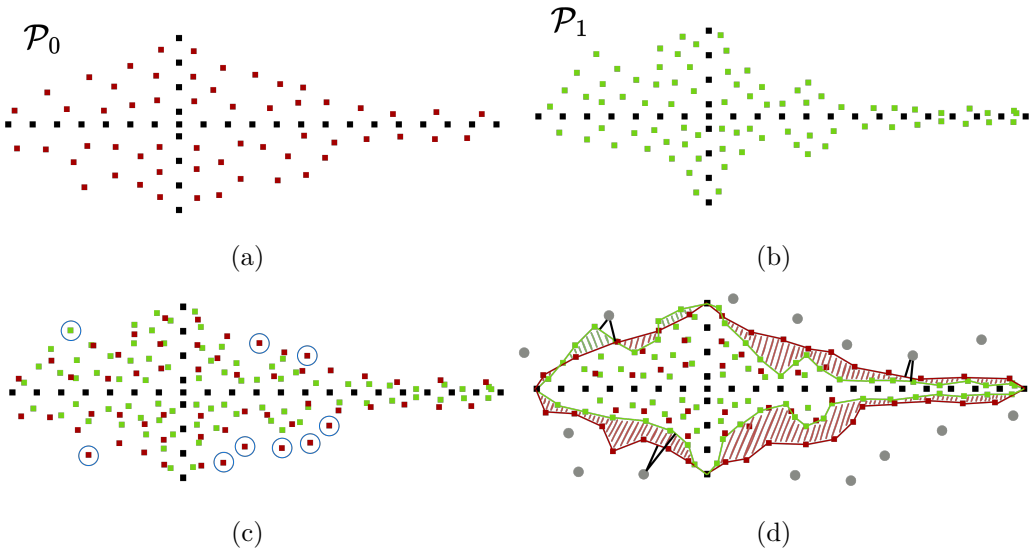


Figure 8.10: (a) and (b) are two leaves generated from the same skeleton. In (c), we show that when using the Chamfer distance, the outliers in the blue circles are the only ones providing a meaningful distance since most of the points are in the same area. In (d), we show our proposed point-to-mesh distance, where we compute the difference between the distances from each gray point to the two meshes.

Table 8.3: Average mean and standard deviation for the chamfer and point-to-mesh distances, computed over 10 trials on 10 leaves generated conditioning the network with the same skeleton.

Generated Data	Chamfer distance [mm]		point-to-mesh distance [mm]	
	mean	std	mean	std
$\mathcal{D}_{\text{LiDiff}}$	6.14	7.53	55.82	57.87
$\mathcal{D}_{\text{ours}}$	5.69	1.69	67.69	17.58

approach and LiDiff. While the chamfer distances have similar means, LiDiff's standard deviation is approx. 4.5 times higher, likely due to the weaker compliance with the skeleton. For the point-to-mesh distance, we see a larger difference in the mean, more than 1 cm, and again the standard deviation of LiDiff is more than 3 times ours.

Given our previously reported results, we think that our approach provides more precise per-leaf ground truths for tuning trait estimation methods. In contrast, LiDiff produces a wider variety of leaves, at the cost of higher leaf trait errors. Since LiDiff is a general-purpose approach for conditioned diffusion, adding specific losses could help the approach follow more closely the input skeleton and improve the results.

8.4 Discussion

Traditional leaf trait estimation heavily relies on manual measurement, which is accurate but also time-consuming, labor-intensive, and difficult to scale. There are several image-based methods for estimating traits such as the number of leaves, but these methods struggle to capture complex geometric properties. In the 3D space, most approaches are rule-based, leveraging expert knowledge to obtain crop-specific models and templates to extract the relevant traits. Learning-based techniques are almost absent due to the scarcity of 3D datasets annotated with per-leaf ground truth traits, which limits their application and robustness.

We propose a method to overcome the challenge of data scarcity for leaf trait estimation. Our generative pipeline outputs realistic 3D leaf point clouds conditioned on the desired traits, i.e., leaf blade length and width. Training on real data, we capture the real-world data distribution without the need for plant-specific parameters or hand-crafted templates. Additionally, our approach provides accurate per-leaf ground truth annotations, resolving the data scarcity problem. This enables tuning the large variety of rule-based methods on a more granular level and opens the road for the development of data-driven methods.

When generating annotated synthetic data, the first aspect to consider is how similar the generated data is to the real-world data, which is crucial to avoid learning features that would not generalize to the final application scenario. The second aspect is the accuracy of the associated annotations. We evaluated both of these aspects in our experimental evaluation. We first showed that tuning different trait estimation methods on our generated data yields more accurate results on the estimation of real-world leaf traits. This already confirms that the data distributions match and that the trait annotations are accurate, as the approaches perform well on unseen real-world data. Then, we confirmed that our generated data is significantly more aligned with the real-world data distribution across multiple metrics and on different crop species. However, the results for leaf trait estimation have been evaluated against per-plot ground truth measurements. This makes it hard to assess the real error on the traits, since we do not have access to the real measure for each leaf. It would be beneficial to obtain per-leaf ground truth measurements to have a more accurate evaluation. For the second part of the experiments, it is clear that, since the generative metrics are based on features extracted by another network, we do not have full control over the quality of the results. Different models output very different results, and it is unclear how much they depend on the quality of the generated data and how much on the gap with the training data of the feature extractor.

Although the results in this chapter are promising, there remains a long way to go before achieving a fully autonomous trait estimation pipeline. First of all,

we focused on only two traits, namely blade length and width, because of the lack of available ground truth data to validate our method. However, leaf angle, area, and shape are also crucial traits that should be evaluated for phenotyping purposes. One limitation of our approach is that, while we do not require highly accurate leaf models or templates, we still need to build a rough skeleton for the network to generate a leaf. This may not be fully generalizable across all crop species, especially for leaves with very complex shapes that may require a more complex skeleton definition when there is not enough data available to learn the distribution around the current simplified skeleton.

8.5 Conclusions

In this chapter, we address the data bottleneck in working with 3D point clouds of leaves by presenting a novel approach that exploits real-world data to generate realistic, trait-conditioned leaf point clouds. Our generated data mirrors the complexity and diversity of natural leaves; thus, we can directly use it for training or tuning off-the-shelf leaf trait estimation approaches that previously struggled due to the limited coarse-grained data.

Our experiments show that optimizing the hyperparameters of different methods on our generated leaf point clouds significantly outperforms the results achieved with traditional plot-level averages and other synthetic state-of-the-art generative methods. By enabling precise, per-leaf trait predictions without requiring any labeling, our data generation pipeline opens the door to fine-grained phenotyping at scale, with direct implications for understanding crop development and productivity. Even when real-world per-leaf ground truth measurements are available, our approach can generate complementary leaves across varying lengths and widths to fill potential gaps in the collected data. This reduces the need for destructive measurements and expert annotation, paving the way for more scalable, unbiased, and informative trait datasets.

Chapter 9

Conclusion

The increasing global population, coupled with the unsustainable nature of conventional farming practices, is placing immense pressure on our agricultural system. The production system must now meet rising demands for food, feed, fuel, and fiber while being more sustainable than ever before. To address this challenge, we must rethink our farming procedures; boosting yield per unit area appears to be one of the main directions to answer all of these challenges.

Robotic technologies present a promising and more sustainable alternative to traditional methods, enabling farmers to adopt high-throughput practices. Unlike conventional equipment, agricultural robots can perform precision tasks and continuous monitoring, significantly reducing the reliance on agrochemicals and supplying valuable data for breeders and agronomists to cultivate more resilient and productive crop varieties. For robots to carry out these tasks effectively, they require advanced perception systems capable of accurate field-level monitoring and measurements. These systems typically rely on data-driven models trained on manually labeled examples. To function reliably, they must be exposed to extensive and diverse datasets that account for variations in plant growth stages, lighting conditions, soil types, and crop species. However, producing such labeled datasets is both time-consuming and costly, creating a major obstacle to the broader adoption of robotic solutions. In the agricultural domain, labeling data is even more burdensome due to the complexity and variability of real-world field conditions. Accurately annotating sensor data often requires domain expertise to distinguish between similar crop species and to mark precise boundaries between adjacent and overlapping plants and leaves. This process is even more labor-intensive when we consider that it should be repeated across multiple seasons, growth stages, crop species, and environmental conditions to ensure the robustness of common fully supervised approaches. Such methods do not generalize well across unseen field conditions and struggle when facing variations in

the data due to sensor setup, weather, soil composition, and vegetation density and species. This lack of adaptability significantly limits the adoption of such models in real-world deployments, since they would require a new round of data collection and labeling to achieve satisfactory performance on the new scenario. Fortunately, agriculture offers unique opportunities to address this limitation. By leveraging existing knowledge about field structures and plant traits, we can enhance robotic perception systems and simultaneously reduce their dependence on large volumes of labeled data.

This thesis directly contributes to making robotic phenotyping more scalable and accessible by developing a set of novel computer vision techniques to reduce the reliance on manual annotations for all the perception tasks in the phenotyping pipeline. We address the understanding of sensor data from agricultural robotic systems to enable high-throughput and precise monitoring of plants at different levels of granularity, covering the entirety of the typical phenotyping pipeline: semantic segmentation, instance segmentation of plants and leaves, and the estimation of morphological leaf traits. The approaches we presented in this work not only allow us to reduce the amount of manually annotated data, but also to completely remove this dependency for semantic segmentation, plant instance segmentation, and leaf trait estimation. Furthermore, all the proposed techniques demonstrate improved generalization capability compared to their fully supervised counterparts across diverse field conditions and crop species, addressing the key limitation that has hindered the broader adoption of robotic solutions in agricultural procedures. The last contribution of this thesis is an approach to generate leaf point clouds along with their associated morphological traits. This can be the starting point for developing data-hungry methods for leaf trait estimation, which were infeasible prior to our work due to the scarcity of single-leaf ground truth trait annotations.

9.1 Summary of the Key Contributions

As a first contribution, we propose a method to jointly identify weeds, crops, individual plants, and single leaves in RGB images from real-world fields. We demonstrate that even a single domain-specific post-processing operation can significantly improve the performance. The results obtained with this approach led us to investigate how we can incorporate domain knowledge to reduce the dependency on labeled data.

The second contribution of this thesis is an approach for task-agnostic pre-training that leverages prior knowledge of the agricultural environment to improve the performance of phenotyping tasks while reducing the need for annotated data. However, pre-training is not enough to perform the desired in-field phenotyping

tasks. Thus, we decided to focus on single tasks to better leverage domain knowledge and eliminate the need for labeled data.

As a third contribution, we present an approach that leverages the spatial arrangement of managed agricultural fields to generate automatic labels for soil-weed-crop segmentation. We then exploit the uncertainty of the network on the less-present class in our automatically labeled data, i.e., the weed class, to refine the network's prediction and bridge the gap with fully supervised methods.

Fourth, proceeding along the phenotyping pipeline, we perform plant instance segmentation without using manually annotated data. We exploit knowledge specific to crops for refining instance proposals generated by foundation models or heuristics-based methods. Without our refinement steps, the investigated approaches suffer because of the overlapping plants and the diversity of crops, growth stages, lighting conditions, and soil textures.

For the fifth contribution, we shift our focus to the 3D domain, using point clouds. This is, in the first place, motivated by our desire to compute leaf morphological traits that are harder to detect in 2D images due to the lack of depth information. We first propose an approach to pre-train 3D neural networks for the leaf instance segmentation task. Together with our proposed automatic post-processing, we leverage the plant morphology to reduce the need for 3D labels and to improve the accuracy of the predictions. Given single leaves, we can now focus on the final step of phenotyping: extracting morphological leaf traits.

As a final contribution, we introduce a method for generating realistic 3D leaf point clouds along with their morphological traits, specifically the leaf blade length and width. Our approach directly addresses the challòenge of the scarcity of fine-grained labeled data for trait estimation, which is also the cause behind the shortage of deep-learning algorithms for trait estimation. Our generative method does not require labeled data, but only realistic leaf point clouds. We use our generated data to validate and optimize heuristics-based trait estimation methods, whose performance is limited when optimized using per-plot averages of the traits. Our method opens the road to the development of deep learning-based models, for which we provide large and accurately labeled synthetic data.

We proposed solutions to reduce reliance on labeled data for all steps in the phenotyping pipeline. We demonstrated how to incorporate prior knowledge to improve the performance across multiple perception tasks, from semantic segmentation to the estimation of leaf morphological traits. Throughout this thesis, we have shown the effectiveness of our proposed solutions on real-world datasets collected in crop fields. This thesis advances the scene understanding of perception systems in the agricultural fields, enabling fine-grained automatic plant monitoring while reducing the cost and labor needed to achieve such performance. While this thesis does not address all the problems and challenges of automatic

in-field phenotyping, the contributions presented are key components to enable autonomous in-field crop monitoring without the need for annotated datasets covering all possible scenarios.

Our contributions meaningfully transform current agricultural systems and offer a concrete direction for more intelligent and sustainable farming practices. The reduced reliance on manual data and the integration of prior knowledge enables the deployment of robust vision-based crop monitoring systems in unseen real-world fields, where data collection is challenging, costly, and labor-intensive. Employing our methods enables the gathering of detailed information about plant health and development in an automatic fashion, allowing for fast and precise targeted intervention in the fields. We provide techniques to enhance all steps in the phenotyping pipeline, enabling their integration with pre-existing heuristic-based or data-driven approaches. As already mentioned, in complex scenarios, it is still possible to use our methods to bootstrap learning-based methods, thus requiring fewer manually labeled examples to achieve better performance.

9.2 Future Work

In this thesis, we presented multiple new methods to leverage prior knowledge of the agricultural domain, thereby reducing the reliance on manually annotated datasets for visual inspection of agricultural fields. Although our methods demonstrate notable improvements over existing state-of-the-art solutions, further research is required to enhance the capabilities of agricultural perception systems without increasing the need for manual annotations.

Multi-Modality. As for the work in this thesis, we developed approaches for both images (2D) and point cloud (3D) data. Each modality, however, has its own challenges and limitations. RGB images provide limited geometric information, which limits their application in accurately separating plant or leaf instances and in estimating phenotypic traits. Point clouds provide more geometric cues, but their dependence on the sensor type and resolution often limits the generalization of trained models across different acquisition setups. Data acquired with LiDAR sensors commonly lacks color information, which is essential for tasks such as disease detection and plant health monitoring.

Although the methods presented in this thesis address many of these limitations and advanced automatic phenotyping in both 2D and 3D, we believe that fusing the strengths of both modalities would further improve the performance of perception tasks for agriculture. The architecture proposed by Jaegle et al. [96] supports multi-modal input and output, enabling the joint use of RGB images and 3D point clouds. This facilitates the fusion of latent information, allowing geometric cues from 3D data to enhance 2D instance segmentation and

trait estimation, and enabling 2D image features to correct 3D predictions when sensor limitations degrade performance. Moreover, the availability of pre-trained models and datasets is currently much greater for 2D images than for 3D point clouds. The architecture by Jaegle et al. [96] could leverage these 2D pre-training resources for 3D tasks, reducing both the time and the amount of data required for effective model training.

Uncertainty Calibration. Modern deep-learning approaches often suffer from poor calibration, meaning that their predicted probabilities do not accurately reflect the true likelihood of outcomes. This issue affects our ability to use uncertainty estimates for refining our semantic prediction in Chapter 5. In particular, overconfident yet incorrect predictions would remain incorrect. This overconfidence arises from multiple factors, including over-parametrization, the use of normalization layers, overfitting, and the optimization procedure. Recent advances aim to improve the uncertainty calibration, yielding a more reliable likelihood for the network's prediction. Since most of the overconfident erroneous predictions happen due to out-of-distribution and uncommon examples, approaches such as the one by Park et al. [167] and Gong et al. [69] could directly address these problems and consequently improve generalization under both domain and covariance shifts.

In Chapter 5, we use an uncertainty-aware training loss to have a better-calibrated uncertainty, but this does not entirely solve the problem. One possible way to integrate current research with our model is to add a post-hoc calibration method to refine the prediction confidence. Most post-hoc uncertainty calibration techniques, such as temperature scaling [13, 56] or isotonic regression [160, 237], were developed for classification tasks, and they often fail to accurately capture pixel-wise uncertainties. As a second possibility, we could follow the previous section of future work and add a second modality, such as point clouds, to enable a cross-modal calibration. Since the use of Bundle Adjustment [207] allows the reconstruction of high-quality point clouds from the sequence of images captured in the field, we could deploy a 3D semantic segmentation network next to our 2D semantic segmentation network and use the uncertainties of the two modalities to have a better estimate of the prediction confidence.

Unsupervised Leaf Instance Segmentation. In Chapter 7, we mentioned that to better align our pre-training with the common paradigm of current instance segmentation networks, we would need to find a self-supervised way to supervise the predictions of centers and offsets. While our leaf instance segmentation was the starting point for trait estimation, the approach presented in Chapter 8 can also be employed to solve the limitation of our previous method. We discussed how we can manipulate the generated leaf point clouds and assemble them into single plants. In doing so, we gain access to a plant point cloud

with leaf instance labels that we generated without using any labels. These plant point clouds can be used as training data for the pre-training approach presented in Chapter 7, obtaining a complete alignment with the final task. On top of that, the fine-tuning step employed to close the gap with the final task and data may be redundant, as we would be able to perform the same task, and we demonstrated that the data generated by the approach in Chapter 8 has high similarity with the real-world data distribution.

Trait Estimation. In Chapter 8, we tested our approach on different crop varieties, all exhibiting similar shape complexity. Our approach also works on more complex shapes, as compound leaves, if enough data is provided. Since the network learns from real-world data, we can use the whole compound leaf as it is and let the network learn its shape. Nonetheless, structural complexity presents additional challenges, and some enhancements could improve the convergence speed and the overall performance. One improvement involves modifying the algorithm for building the skeleton. For example, we could combine multiple skeletons to capture the morphology of a compound leaf. Adjusting the number of Gaussians J in the GMM can improve the initial position of the points, leading to more efficient training. While these changes are not strictly required, they could diminish the need for data and provide a better and more effective initialization for complex leaf shapes. Another potential direction is to learn the shape of single leaflets instead of the entire compound leaf. This method wouldn't require algorithmic changes, but it would require access to single leaflet point clouds – harder to obtain than single leaves. Furthermore, knowledge about the leaf structure would be needed to reconstruct complete leaves from the generated leaflets via a post-processing step.

A second interesting direction would be the application of our method to generate distinct varieties within a single crop species. This would require large, variety-specific datasets, as the network must learn finer morphological details. Currently, the primary limitation is the lack of available data. Moreover, existing evaluation metrics often rely on networks pre-trained on large datasets of common objects [31, 51], which may struggle to differentiate between single crop species varieties. A domain-specific foundation model tailored to plant data would yield more reliable evaluations.

Additionally, we think our method would benefit from being combined with plant growth models. Since our network can be trained on leaves from specific growth stages, we can generate a large variety of leaves for each stage. Plant growth models could provide the expected leaf blade length and width based on environmental and nutritional input, allowing for more realistic simulations without the need for external user-defined inputs.

Bibliography

[1] M. Abdar, F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghavamzadeh, P. Fieguth, X. Cao, A. Khosravi, U.R. Acharya, et al. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 76:243–297, 2021.

[2] I. Achituve, H. Maron, and G. Chechik. Self-supervised learning for domain adaptation on point clouds. In *Proc. of the IEEE Winter Conf. on Applications of Computer Vision (WACV)*, 2021.

[3] J. Agrawal and M.Y. Arafat. Transforming farming: A review of ai-powered uav technologies in precision agriculture. *Drones*, 8(11), 2024.

[4] A. Ahmadi, M. Halstead, and C. McCool. Virtual temporal samples for recurrent neural networks: Applied to semantic segmentation in agriculture. *Pattern Recognition*, pages 574–588, 2021.

[5] A. Ahmadi, M. Halstead, and C. McCool. BonnBot-I: A Precise Weed Management and Crop Monitoring Platform. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2022.

[6] S.K. Aithal, P. Maini, Z. Lipton, and J.Z. Kolter. Understanding hallucinations in diffusion models through mode interpolation. *Advances in Neural Information Processing Systems*, 37:134614–134644, 2024.

[7] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proc. of the Intl. Conf. on Knowledge Discovery and Data Mining (KDD)*, 2019.

[8] A. Alliegro, D. Boscaini, and T. Tommasi. Joint supervised and self-supervised learning for 3d real world challenges. In *Proc. of the Intl. Conf. on Pattern Recognition (ICPR)*, 2021.

[9] Z. Ao, F. Wu, S. Hu, Y. Sun, Y. Su, Q. Guo, and Q. Xin. Automatic segmentation of stem and leaf components and individual maize plants in field terrestrial LiDAR data using convolutional neural networks. *The Crop Journal*, 10(5):1239–1250, 2022.

[10] G.P. Asner, J.M.O. Scurlock, and J.A. Hicke. Global synthesis of leaf area index observations: implications for ecological and remote sensing studies. *Global Ecology and Biogeography*, 12(3):191–205, 2003.

[11] B.N. Bailey. Helios: A scalable 3d plant and environmental biophysical modeling framework. *Frontiers in Plant Science*, 10:1185, 2019.

[12] A. Balabantaray, S. Behera, C. Liew, N. Chamara, M. Singh, A.J. Jhala, and S. Pitla. Targeted weed management of palmer amaranth using robotics and deep learning (yolov7). *Frontiers in Robotics and AI*, 11, 2024.

[13] S.A. Balanya, J. Maroñas, and D. Ramos. Adaptive temperature scaling for robust calibration of deep neural networks. *Neural Computing and Applications*, 36(14):8073–8095, 2024.

[14] M. Bayati and R. Fotouhi. A mobile robotic platform for crop monitoring. *Advances in Robotics & Automation*, 7(1):1000186, 2018.

[15] W.H. Beluch, T. Genewein, A. Nürnberger, and J.M. Köhler. The Power of Ensembles for Active Learning in Image Classification. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[16] J.Z. Bengar, J. van de Weijer, B. Twardowski, and B. Raducanu. Reducing label effort: Self-supervised meets active learning. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[17] M. Berman, A.R. Triki, and M.B. Blaschko. The Lovász-Softmax Loss: A Tractable Surrogate for the Optimization of the Intersection-Over-Union Measure in Neural Networks. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[18] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Trans. on Visualization and Computer Graphics*, 5(4):349–359, 1999.

[19] M. Bińkowski, D.J. Sutherland, M. Arbel, and A. Gretton. Demystifying MMD GANs. In *Proc. of the Intl. Conf. on Learning Representations (ICLR)*, 2018.

[20] P. Bosilj, E. Aptoula, T. Duckett, and G. Cielniak. Transfer learning between crop types for semantic segmentation of crops versus weeds in precision agriculture. *Journal of Field Robotics (JFR)*, 37(1):7–19, 2020.

[21] M. Boukhana, J. Ravaglia, F. Hétroy-Wheeler, and B. De Solan. Geometric models for plant leaf area estimation from 3D point clouds: A comparative study. *Graphics and Visual Computing*, 7:200057, 2022.

[22] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 120:122–125, 2000.

[23] C.R. Brice and C.L. Fennema. Scene analysis using regions. *Artificial Intelligence*, 1(3-4):205–226, 1970.

[24] R.J. Campello, D. Moulavi, and J. Sander. Density-Based Clustering Based on Hierarchical Density Estimates. In *Proc. of the Conf. on Knowledge Discovery and Data Mining (KDD)*, 2013.

[25] J.F. Canny. A computational approach to edge detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, 8(12):679–698, 1986.

[26] H. Cao, Y. Wang, J. Chen, D. Jiang, X. Zhang, Q. Tian, and M. Wang. Swin-unet: Unet-like pure transformer for medical image segmentation. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2023.

[27] C. Carbone, C. Potena, and D. Nardi. Simulation of near infrared sensor in unity for plant-weed segmentation classification. In *Proc. of the Intl. Conf. on Simulation and Modeling Methodologies, Technologies and Applications*, 2020.

[28] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *Proc. of the Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

[29] V. Caselles, R. Kimmel, and G. Sapiro. Geodesic active contours. *Intl. Journal of Computer Vision (IJCV)*, 22:61–79, 1997.

[30] J. Champ, A. Mora-Fallas, H. Goëau, E. Mata-Montero, P. Bonnet, and A. Joly. Instance segmentation for the fine detection of crop and weed plants by precision agricultural robots. *Applications in Plant Sciences*, 8(7):e11373, 2020.

[31] A. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University, 2015.

[32] N. Chebrolu, P. Lottes, A. Schaefer, W. Winterhalter, W. Burgard, and C. Stachniss. Agricultural Robot Dataset for Plant Classification, Localization and Mapping on Sugar Beet Fields. *Intl. Journal of Robotics Research (IJRR)*, 36(10):1045–1052, 2017.

[33] D. Chen, J.M. Mirebeau, H. Shu, and L.D. Cohen. A region-based randers geodesic approach for image segmentation. *Intl. Journal of Computer Vision (IJCV)*, 132(2):349–391, 2024.

[34] L. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking Atrous Convolution for Semantic Image Segmentation. *arXiv preprint*, arXiv:1706.05587, 2017.

[35] Q. Chen and X. Qi. Residual Graph Convolutional Network for Bird's-Eye-View Semantic Segmentation. In *Proc. of the IEEE Winter Conf. on Applications of Computer Vision (WACV)*, 2024.

[36] S. Chen, Y. Guo, X. Sirault, K. Stefanova, R. Saradadevi, N.C. Turner, M.N. Nelson, R.T. Furbank, K.H. Siddique, and W.A. Cowling. Nondestructive phenomic tools for the prediction of heat and drought tolerance at anthesis in brassica species. *Plant Phenomics*, 2019.

[37] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A Simple Framework for Contrastive Learning of Visual Representations. In *Proc. of the Intl. Conf. on Machine Learning (ICML)*, 2020.

[38] X. Chen, H. Fan, R. Girshick, and K. He. Improved baselines with momentum contrastive learning. *arXiv preprint*, 2003.04297, 2020.

[39] B. Cheng, M.D. Collins, Y. Zhu, T. Liu, T.S. Huang, H. Adam, and L.C. Chen. Panoptic-DeepLab: A Simple, Strong, and Fast Baseline for Bottom-Up Panoptic Segmentation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[40] B. Cheng, R. Girshick, P. Dollár, A.C. Berg, and A. Kirillov. Boundary iou: Improving object-centric image segmentation evaluation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[41] B. Cheng, I. Misra, A.G. Schwing, A. Kirillov, and R. Girdhar. Masked-attention mask transformer for universal image segmentation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[42] B. Cheng, A.G. Schwing, and A. Kirillov. Per-pixel classification is not all you need for semantic segmentation. In *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 2021.

[43] F. Chollet. Xception: Deep Learning With Depthwise Separable Convolutions. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[44] J.D. Colorado, F.C. Calderon, D. Mendez, E. Petro, J.P. Rojas, E.S. Correa, I.F. Mondragón, M.C. Rebolledo, and A. Jaramillo-Botero. A novel nir-image segmentation method for the precise estimation of above-ground biomass in rice crops. *PLOS ONE*, 15(10):e0239591, 2020.

[45] D. Comaniciu and P. Meer. Mean Shift: A Robust Approach Toward Feature Space Analysis. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, 24(5):603–619, 2002.

[46] F.A. Croitoru, V. Hondru, R.T. Ionescu, and M. Shah. Diffusion models in vision: A survey. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, 45(9):10850–10869, 2023.

[47] G. Csurka, R. Volpi, B. Chidlovskii, et al. Semantic image segmentation: Two decades of research. *Foundations and Trends in Computer Graphics and Vision*, 14(1-2):1–162, 2022.

[48] J. Cui, F. Tan, N. Bai, and Y. Fu. Improving u-net network for semantic segmentation of corns and weeds during corn seedling stage in field. *Frontiers in Plant Science*, 15, 2024.

[49] T. Dao, A. Gu, A. Ratner, V. Smith, C. De Sa, and C. Re. A kernel theory of modern data augmentation. In *Proc. of the Intl. Conf. on Machine Learning (ICML)*, pages 1528–1537, 2019.

[50] S. Das Choudhury, S. Maturu, A. Samal, V. Stoerger, and T. Awada. Leveraging image analysis to compute 3d plant phenotypes based on voxel-grid plant reconstruction. *Frontiers in Plant Science*, 11:521431, 2020.

[51] M. Deitke, D. Schwenk, J. Salvador, L. Weihs, O. Michel, E. VanderBilt, L. Schmidt, K. Ehsani, A. Kembhavi, and A. Farhadi. Objaverse: A universe of annotated 3d objects. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[52] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[53] F. Deravi and S. Pal. Grey level thresholding using second-order statistics. *Pattern Recognition Letters*, 1(5):417–422, 1983.

[54] R.P. Devanna, L. Romeo, G. Reina, and A. Milella. Yield estimation in precision viticulture by combining deep segmentation and depth-based clustering. *Computers and Electronics in Agriculture*, 232:110025, 2025.

[55] T. DeVries and G.W. Taylor. Improved Regularization of Convolutional Neural Networks with Cutout. *arXiv preprint*, arXiv:1708.04552, 2017.

[56] Z. Ding, X. Han, P. Liu, and M. Niethammer. Local temperature scaling for probability calibration. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2021.

[57] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *Proc. of the Intl. Conf. on Learning Representations (ICLR)*, 2021.

[58] D. Erhan, Y. Bengio, A. Courville, P.A. Manzagol, P. Vincent, and S. Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(19):625–660, 2010.

[59] M. Ester, H. Kriegel, J. Sander, and X.Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. of the Conf. on Knowledge Discovery and Data Mining (KDD)*, 1996.

[60] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *Intl. Journal of Computer Vision (IJCV)*, 88(2):303–338, 2010.

[61] G. Farjon, Y. Itzhaky, F. Khoroshevsky, and A. Bar-Hillel. Leaf counting: Fusing network components for improved accuracy. *Frontiers in Plant Science*, 12:575751, 2021.

[62] P.F. Felzenszwalb and D.P. Huttenlocher. Efficient graph-based image segmentation. *Intl. Journal of Computer Vision (IJCV)*, 59(2):167–181, 2004.

[63] R.W. Floyd. Algorithm 97: Shortest path. *Communications of the ACM*, 5:345, 1962.

[64] P. Franti, O. Virmajoki, and V. Hautamaki. Fast agglomerative clustering using a k-nearest neighbor graph. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, 28(11):1875–1881, 2006.

[65] Y. Gal and Z. Ghahramani. Dropout as a Bayesian Approximation: Representing model uncertainty in deep learning. In *Proc. of the Intl. Conf. on Machine Learning (ICML)*, 2016.

[66] P. Ge, C.X. Ren, X.L. Xu, and H. Yan. Unsupervised domain adaptation via deep conditional adaptation network. *Pattern Recognition*, 134:109088, 2023.

[67] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proc. of the Intl. Conf. on Artificial Intelligence and Statistics*, 2010.

[68] H. Godfray, J. Beddington, R. Crute, L. Haddad, D. Lawrence, J. Muir, J. Pretty, S. Robinson, S. Thomas, and C. Toulmin. Food security: the challenge of feeding 9 billion people. *Science*, 327 5967:812–8, 2010.

[69] Y. Gong, X. Lin, Y. Yao, T.G. Dietterich, A. Divakaran, and M. Gervasio. Confidence calibration for domain generalization under covariate shift. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[70] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Networks. In *Proc. of the Conf. Neural Information Processing Systems (NIPS)*, 2014.

[71] J.B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, B. Piot, k. kavukcuoglu, R. Munos, and M. Valko. Bootstrap Your Own Latent - A New Approach to Self-Supervised Learning. In *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 2020.

[72] A. Gupta, P. Vuillecard, A. Farkhondeh, and J.M. Odobez. Exploring the zero-shot capabilities of vision-language models for improving gaze following. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[73] F. Görlich, E. Marks, A.K. Mahlein, K. König, P. Lottes, and C. Stachniss. UAV-Based Classification of Cercospora Leaf Spot Using RGB Images. *Drones*, 5(2):34, 2021.

[74] R. Güldenring, R.E. Andersen, and L. Nalpantidis. Zoom in on the plant: Fine-grained analysis of leaf, stem, and vein instances. *IEEE Robotics and Automation Letters (RA-L)*, 9(2):1588–1595, 2024.

[75] A.M. Hafiz and G.M. Bhat. A survey on instance segmentation: state of the art. *Intl. Journal of Multimedia Information Retrieval*, 9(3):171–189, 2020.

[76] M. Halstead, A. Ahmadi, C. Smitt, O. Schmittmann, and C. McCool. Crop agnostic monitoring driven by deep learning. *Frontiers in Plant Science*, 12, 2021.

[77] M. Hamilton, Z. Zhang, B. Hariharan, N. Snavely, and W.T. Freeman. Unsupervised semantic segmentation by distilling feature correspondences. In *Proc. of the Intl. Conf. on Machine Learning (ICML)*, 2022.

[78] B. Han, Y. Li, Z. Bie, C. Peng, Y. Huang, and S. Xu. MIX-NET: Deep Learning-Based Point Cloud Processing Method for Segmentation and Occlusion Leaf Restoration of Seedlings. *Plants*, 11(23):3342, 2022.

[79] G. Han and S.N. Lim. Few-shot object detection with foundation models. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[80] P.E. Hart, N.J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[81] M. Hassanein, Z. Lari, and N. El-Sheimy. A new vegetation segmentation approach for cropped fields based on threshold detection from hue histograms. *Sensors*, 18(4):1253, 2018.

[82] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum Contrast for Unsupervised Visual Representation Learning. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[83] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, 2017.

[84] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[85] D. Hegde, J.M.J. Valanarasu, and V. Patel. Clip goes 3d: Leveraging prompt tuning for language grounded 3d recognition. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[86] D.N. Helmrich, F.M. Bauer, M. Giraud, A. Schnepf, J.H. Göbbert, H. Scharr, E.T. Hvannberg, and M. Riedel. A scalable pipeline to create synthetic datasets from functional–structural plant models for deep learning. *In Silico Plants*, 6(1):diad022, 2024.

[87] D. Hendrycks and K. Gimpel. Gaussian Error Linear Units (GELUs). *arXiv preprint*, arXiv:1606.08415, 2016.

[88] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *Proc. of the Conf. Neural Information Processing Systems (NIPS)*, 2017.

[89] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 30, 2017.

[90] P.V.C. Hough. Machine analysis of bubble chamber pictures. In *Proc. of the Intl. Conf. on High-Energy Accelerators and Instrumentation*, 1959.

[91] C.C. Hsu, K.J. Hsu, C.C. Tsai, Y.Y. Lin, and Y.Y. Chuang. Weakly supervised instance segmentation using the bounding box tightness prior. *Advances in Neural Information Processing Systems*, 32, 2019.

[92] G. Huang, Z. Liu, L. Maaten, and K.Q. Weinberger. Densely Connected Convolutional Networks. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[93] X. Huang, S. Zheng, and L. Gui. Automatic measurement of morphological traits of typical leaf samples. *Sensors*, 21(6):2247, 2021.

[94] M. Höffmann, S. Patel, and C. Büskens. Optimal coverage path planning for agricultural vehicles with curvature constraints. *Agriculture*, 13(11):2112, 2023.

[95] J. Iqbal, R. Xu, S. Sun, and C. Li. Simulation of an autonomous mobile robot for lidar-based in-field phenotyping and navigation. *Robotics*, 9(2):46, 2020.

[96] A. Jaegle, S. Borgeaud, J.B. Alayrac, C. Doersch, C. Ionescu, D. Ding, S. Koppula, D. Zoran, A. Brock, E. Shelhamer, et al. Perceiver io: A general architecture for structured inputs & outputs. In *Proc. of the Intl. Conf. on Learning Representations (ICLR)*, 2022.

[97] S. Jayasumana, S. Ramalingam, A. Veit, D. Glasner, A. Chakrabarti, and S. Kumar. Rethinking fid: Towards a better evaluation metric for image generation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[98] W.S. Jeon, G. Cielniak, and S.Y. Rhee. Semantic segmentation using trade-off and internal ensemble. *Intl. Journal of Fuzzy Logic and Intelligent Systems*, 18(3):196–203, 2018.

[99] W. Jia, J. Liu, Y. Lu, Q. Liu, T. Zhang, and X. Dong. Polar-net: Green fruit instance segmentation in complex orchard environment. *Frontiers in Plant Science*, 13, 2022.

[100] H. Jiang, F. Yan, J. Cai, J. Zheng, and J. Xiao. End-to-end 3D Point Cloud Instance Segmentation without Detection. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[101] H. Jiang, J. Jang, and S. Kpotufe. Quickshift++: Provably good initializations for sample-based mean shift. In *Proc. of the Intl. Conf. on Machine Learning (ICML)*, 2018.

[102] S. Jin, Y. Su, F. Wu, S. Pang, S. Gao, T. Hu, J. Liu, and Q. Guo. Stem–leaf segmentation and phenotypic trait extraction of individual maize using terrestrial LiDAR data. *IEEE Trans. on Geoscience and Remote Sensing*, 57(3):1336–1346, 2018.

[103] L. Jing and Y. Tian. Self-supervised visual feature learning with deep neural networks: A survey. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, 43(11):4037–4058, 2020.

[104] A. Kamilaris and F.X. Prenafeta-Boldú. Deep learning in agriculture: A survey. *Computers and Electronics in Agriculture*, 147:70–90, 2018.

[105] G. Karypis and V. Kumar. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998.

[106] J. Kierdorf, L.V. Junker-Frohn, M. Delaney, M.D. Olave, A. Burkart, H. Jaenicke, O. Muller, U. Rascher, and R. Roscher. Growliflower: An image time series dataset for growth analysis of cauliflower. *Journal of Field Robotics (JFR)*, 40(2):173–192, 2022.

[107] D. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *Proc. of the Intl. Conf. on Learning Representations (ICLR)*, 2015.

[108] A. Kirillov, R. Girshick, K. He, and P. Dollar. Panoptic Feature Pyramid Networks. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[109] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár. Panoptic Segmentation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[110] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A.C. Berg, W.Y. Lo, P. Dollár, and R. Girshick. Segment anything. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[111] J. Kittler and J. Illingworth. On threshold selection using clustering criteria. *IEEE Trans. on Systems, Man, and Cybernetics*, 15(5):652–655, 1985.

[112] O. Knopf, A. Castro, J. Bendig, R. Pude, E. Kleist, H. Poorter, U. Rascher, and O. Muller. Field phenotyping of ten wheat cultivars under elevated co2 shows seasonal differences in chlorophyll fluorescence, plant height and vegetation indices. *Frontiers in Plant Science*, 14, 2024.

[113] S. Kolhar and J. Jagtap. Plant trait estimation and classification studies in plant phenotyping using machine vision–a review. *Information Processing in Agriculture*, 10(1):114–135, 2023.

[114] M. Krichen. Generative adversarial networks. In *Proc. of Intl. Conf. on Computing Communication and Networking Technologies (ICCCNT)*, pages 1–7, 2023.

[115] T. Kynkäänniemi, T. Karras, S. Laine, J. Lehtinen, and T. Aila. Improved precision and recall metric for assessing generative models. In *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 2019.

[116] Q. Lai, C.M. Vong, and C. Chen. Weakly supervised semantic segmentation via dual-stream contrastive learning of cross-image contextual information. *IEEE Trans. on Industrial Informatics*, 2024.

[117] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and Scalable Predictive Uncertainty Estimation Using Deep Ensembles. In *Proc. of the Conf. Neural Information Processing Systems (NIPS)*, 2017.

[118] P.D. Lancashire, H. Bleiholder, T.V.D. Boom, P. Langelüddeke, R. Stauss, E. Weber, and A. Witzenberger. A uniform decimal code for growth stages of crops and weeds. *Annals of Applied Biology*, 119(3):561–601, 1991.

[119] S. Lawrence, C.L. Giles, and A.C. Tsoi. What size neural network gives optimal generalization? convergence properties of backpropagation. Technical Report 3617, University of Maryland Institute for Advanced Computer Studies, 1998.

[120] P. Leonetti, M.S. Hanafy, R. Tayade, M. Ramakrishnan, H. Sonah, and H.J. Jacobsen. Leveraging genomics, phenomics, and plant biotechnology approaches for improving abiotic and biotic stress tolerance in cereals and legumes. *Frontiers in Plant Science*, 14:1307390, 2023.

[121] C. Li, Z. Gan, Z. Yang, J. Yang, L. Li, L. Wang, J. Gao, et al. Multimodal foundation models: From specialists to general-purpose assistants. *Foundations and Trends in Computer Graphics and Vision*, 16(1-2):1–214, 2024.

[122] D. Li, J. Li, S. Xiang, and A. Pan. Psegnet: Simultaneous semantic and instance segmentation for point clouds of plants. *Plant Phenomics*, 2022:9787643, 2022.

[123] D. Li, G. Shi, J. Li, Y. Chen, S. Zhang, S. Xiang, and S. Jin. Plantnet: A dual-function point cloud segmentation network for multiple plant species. *ISPRS Journal of Photogrammetry and Remote Sensing (JPRS)*, 184:243–263, 2022.

[124] T. Li, J. Burridge, P.M. Blok, and W. Guo. A patch-level data synthesis pipeline enhances species-level crop and weed segmentation in natural agricultural scenes. *Agriculture*, 15(2), 2025.

[125] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen. PointCNN: Convolution On X-Transformed Points. In *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 2018.

[126] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen. Pointcnn: Convolution on x-transformed points. *Advances in Neural Information Processing Systems*, 31, 2018.

[127] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C.L. Zitnick. Microsoft COCO: Common Objects in Context. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2014.

[128] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, Q. Jiang, C. Li, J. Yang, H. Su, J. Zhu, and L. Zhang. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2025.

[129] Z. Liu, H. Mao, C.Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie. A convnet for the 2020s. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[130] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *Proc. of the Intl. Conf. on Learning Representations (ICLR)*, 2019.

[131] P. Lottes, J. Behley, N. Chebrolu, A. Milioto, and C. Stachniss. Robust Joint Stem Detection and Crop-Weed Classification using Image Sequences for Plant-Specific Treatment in Precision Farming. *Journal of Field Robotics (JFR)*, 37(1):20–34, 2020.

[132] P. Lottes, M. Höferlin, S. Sander, M. Müter, P. Schulze-Lammers, and C. Stachniss. An Effective Classification System for Separating Sugar Beets and Weeds for Precision Farming Applications. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2016.

[133] P. Lottes, M. Höferlin, S. Sander, and C. Stachniss. Effective Vision-based Classification for Separating Sugar Beets and Weeds for Precision Farming. *Journal of Field Robotics (JFR)*, 34(6):1160–1178, 2017.

[134] P. Lottes and C. Stachniss. Semi-supervised online visual crop and weed classification in precision farming exploiting plant arrangement. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2017.

[135] S.P. Loyd. Least squares quantization in pcm. *IEEE Trans. on Information Theory*, 28(2):129–137, 1982.

[136] L. Luo, X. Jiang, Y. Yang, E.R.A. Samy, M. Lefsrud, V. Hoyos-Villegas, and S. Sun. Eff-3dpseg: 3d organ-level plant shoot segmentation using annotation-efficient deep learning. *Plant Phenomics*, 5:0080, 2023.

[137] A. Lyasmine, F. Idir, and B. Samia. Plant leaf image segmentation in natural scenes: a multi-layer graph queries propagation approach. *Pattern Analysis and Applications*, 28(1):1–23, 2025.

[138] F. Magistri, N. Chebrolu, and C. Stachniss. Segmentation-Based 4D Registration of Plants Point Clouds for Phenotyping. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2020.

[139] F. Magistri, R. Marcuzzi, E. Marks, M. Sodano, J. Behley, and C. Stachniss. Efficient and Accurate Transformer-Based 3D Shape Completion and Reconstruction of Fruits for Agricultural Robots. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2024.

[140] R. Mahmood, J. Lucas, D. Acuna, D. Li, J. Philion, J.M. Alvarez, Z. Yu, S. Fidler, and M.T. Law. How much more data do i need? estimating requirements for downstream tasks. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[141] E. Marks, F. Magistri, and C. Stachniss. Precise 3D Reconstruction of Plants from UAV Imagery Combining Bundle Adjustment and Template Matching. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2022.

[142] E. Marks, J. Bömer, F. Magistri, A. Sah, J. Behley, and C. Stachniss. BonnBeetClouds3D: A Dataset Towards Point Cloud-Based Organ-Level Phenotyping of Sugar Beet Plants Under Real Field Conditions. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2024.

[143] E. Marks, M. Sodano, F. Magistri, L. Wiesmann, D. Desai, R. Marcuzzi, J. Behley, and C. Stachniss. High precision leaf instance segmentation for phenotyping in point clouds obtained under real field conditions. *IEEE Robotics and Automation Letters (RA-L)*, 8(8):4791–4798, 2023.

[144] T. Masuda. Leaf area estimation by semantic segmentation of point cloud of tomato plants. In *Proc. of the Intl. Conf. on Computer Vision Workshops*, 2021.

[145] C. McCool, T. Perez, and B. Upcroft. Mixtures of Lightweight Deep Convolutional Neural Networks: Applied to Agricultural Robotics. *IEEE Robotics and Automation Letters (RA-L)*, 2(3):1344–1351, 2017.

[146] R. Menon, T. Zaenker, N. Dengler, and M. Bennewitz. NBV-SC: Next Best View Planning based on Shape Completion for Fruit Mapping and Reconstruction. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2023.

[147] A.T. Meshram, A.V. Vanalkar, K.B. Kalambe, and A.M. Badar. Pesticide spraying robot for precision agriculture: A categorical literature review and future trends. *Journal of Field Robotics (JFR)*, 39(2):153–171, 2022.

[148] T. Miao, C. Zhu, T. Xu, T. Yang, N. Li, Y. Zhou, and H. Deng. Automatic stem-leaf segmentation of maize shoots using three-dimensional point cloud. *Computers and Electronics in Agriculture*, 187:106310, 2021.

[149] Z. Migicovsky, M. Li, D.H. Chitwood, and S. Myles. Morphometrics reveals complex and heritable apple leaf shapes. *Frontiers in Plant Science*, 8, 2018.

[150] A. Milioto, P. Lottes, and C. Stachniss. Real-time Semantic Segmentation of Crop and Weed for Precision Agriculture Robots Leveraging Background Knowledge in CNNs. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2018.

[151] A. Milioto, L. Mandtler, and C. Stachniss. Fast Instance and Semantic Segmentation Exploiting Local Connectivity, Metric Learning, and One-Shot Detection for Robotics. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2019.

[152] M. Minervini, H. Scharr, and S. Tsaftaris. Image analysis: The new bottleneck in plant phenotyping. *IEEE Signal Processing Magazine*, 32:126–131, 2015.

[153] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. In *Proc. of the Intl. Conf. on Learning Representations (ICLR)*, 2018.

[154] S.S. Mohammadi, Y. Wang, and A. Del Bue. Pointview-gcn: 3d shape classification with multi-view point clouds. In *Proc. of the IEEE Intl. Conf. on Image Processing (ICIP)*, 2021.

[155] D. Morris. A Pyramid CNN for Dense-Leaves Segmentation. In *Proc. of the Conf. on Computer and Robot Vision (CRV)*, 2018.

[156] K. Murugan, B.J. Shankar, A. Sumanth, C.V. Sudharshan, and G.V. Reddy. Smart automated pesticide spraying bot. In *Proc. of the Intl. Conf. on Intelligent Sustainable Systems (ICISS)*, 2020.

[157] A. Myronenko and X. Song. Point set registration: Coherent point drift. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, 32(12):2262–2275, 2010.

[158] L. Najman and M. Schmitt. Geodesic saliency of watershed contours and hierarchical segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, 18(12):1163–1173, 1996.

[159] H.J. Nelson and N. Papanikolopoulos. Pre-clustering point clouds of crop fields using scalable methods. *arXiv preprint*, arXiv:2107.10950, 2021.

[160] A. Niculescu-Mizil and R. Caruana. Predicting good probabilities with supervised learning. In *Proc. of the Intl. Conf. on Machine Learning (ICML)*, 2005.

[161] D. Niu, X. Wang, X. Han, L. Lian, R. Herzig, and T. Darrell. Unsupervised universal image segmentation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[162] L. Nunes, R. Marcuzzi, X. Chen, J. Behley, and C. Stachniss. SegContrast: 3D Point Cloud Feature Representation Learning through Self-supervised Segment Discrimination. *IEEE Robotics and Automation Letters (RA-L)*, 7(2):2116–2123, 2022.

[163] L. Nunes, R. Marcuzzi, B. Mersch, J. Behley, and C. Stachniss. Scaling Diffusion Models to Real-World 3D LiDAR Scene Completion. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[164] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Trans. on Systems, Man, and Cybernetics*, 9(1):62–66, 1979.

[165] Y. Ozaki, Y. Tanigaki, S. Watanabe, and M. Onishi. Multiobjective tree-structured parzen estimator for computationally expensive optimization problems. In *Proc. of Genetic and Evolutionary Computation Conf. (GECCO)*, 2020.

[166] Y. Pan, F. Magistri, T. Läbe, E. Marks, C. Smitt, C. McCool, J. Behley, and C. Stachniss. Panoptic mapping with fruit completion and pose estimation for horticultural robots. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2023.

[167] S. Park, O. Bastani, J. Weimer, and I. Lee. Calibrated prediction with covariate shift via unsupervised domain adaptation. In *Proc. of the Intl. Conf. on Artificial Intelligence and Statistics*, 2020.

[168] S. Paulus. Measuring crops in 3D: using geometry for plant phenotyping. *Plant Methods*, 15(1):1–13, 2019.

[169] R. Polly and E.A. Devi. Semantic segmentation for plant leaf disease classification and damage detection: A deep learning approach. *Smart Agricultural Technology*, 9:100526, 2024.

[170] T.C. Pong, L.G. Shapiro, L.T. Watson, and R.M. Haralick. Experiments in segmentation using a facet model region grower. *Computer Vision, Graphics, and Image Processing*, 25(1):1–23, 1984.

[171] H. Poorter, G.M. Hummel, K.A. Nagel, F. Fiorani, P. von Gillhaussen, O. Virnich, U. Schurr, J.A. Postma, R. van de Zedde, and A. Wiese-

Klinkenberg. Pitfalls and potential of high-throughput plant phenotyping platforms. *Frontiers in Plant Science*, 14, 2023.

[172] A. Pretto, S. Aravecchia, W. Burgard, N. Chebrolu, C. Dornhege, T. Falck, F. Fleckenstein, A. Fontenla, M. Imperoli, R. Khanna, F. Liebisch, P. Lottes, A. Milioto, D. Nardi, S. Nardi, J. Pfeifer, M. Popović, C. Potena, C. Pradalier, E. Rothacker-Feder, I. Sa, A. Schaefer, R. Siegwart, C. Stachniss, A. Walter, W. Winterhalter, X. Wu, and J. Nieto. Building an Aerial-Ground Robotics System for Precision Farming. *IEEE Robotics and Automation Magazine (RAM)*, 28(3):29–49, 2020.

[173] T. Pun. Entropic thresholding, a new approach. *Computer Graphics and Image Processing*, 16(3):210–239, 1981.

[174] M.A. Rahman and Y. Wang. Optimizing Intersection-Over-Union in Deep Neural Networks for Image Segmentation. In *Proc. of the Intl. Symp. on Visual Computing*, 2016.

[175] N. Ravi, V. Gabeur, Y.T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, E. Mintun, J. Pan, K.V. Alwala, N. Carion, C.Y. Wu, R. Girshick, P. Dollár, and C. Feichtenhofer. Sam 2: Segment anything in images and videos. *arXiv preprint*, arXiv:2408.00714, 2024.

[176] A.S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. CNN Features off-the-shelf: an Astounding Baseline for Recognition. In *Proc. of the CVPR Workshops*, 2014.

[177] S.S. Reddi, S.F. Rudin, and H.R. Keshavan. An optimal multiple threshold scheme for image segmentation. *IEEE Trans. on Systems, Man, and Cybernetics*, 14(4):661–665, 1984.

[178] T. Ren, S. Liu, A. Zeng, J. Lin, K. Li, H. Cao, J. Chen, X. Huang, Y. Chen, F. Yan, et al. Grounded sam: Assembling open-world models for diverse visual tasks. *arXiv preprint*, arXiv:2401.14159, 2024.

[179] D.A. Reynolds. Gaussian mixture models. In *Encyclopedia of Biometrics*, 2018.

[180] J. Rodriguez-Vazquez, M. Fernandez-Cortizas, D. Perez-Saura, M. Molina, and P. Campoy. Overcoming domain shift in neural networks for accurate plant counting in aerial images. *Remote Sensing*, 15(6), 2023.

[181] G. Roggiolani, J.Rückin, M. Popović, J. Behley, and C. Stachniss. Unsupervised Semantic Label Generation in Agricultural Fields. *Frontiers in Robotics and AI*, 12(1):1548143, 2025.

[182] G. Roggiolani, F. Magistri, T. Guadagnino, J. Behley, and C. Stachniss. Unsupervised Pre-Training for 3D Leaf Instance Segmentation. *IEEE Robotics and Automation Letters (RA-L)*, 8(11):7448–7455, 2023.

[183] G. Roggiolani, M. Sodano, F. Magistri, T. Guadagnino, J. Behley, and C. Stachniss. Hierarchical Approach for Joint Semantic, Plant Instance, and Leaf Instance Segmentation in the Agricultural Domain. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2023.

[184] E. Romera, J.M. Alvarez, L.M. Bergasa, and R. Arroyo. ERFNet: Efficient Residual Factorized ConvNet for Real-Time Semantic Segmentation. *IEEE Trans. on Intelligent Transportation Systems (TITS)*, 19(1):263–272, 2018.

[185] B. Romera-Paredes and P. Torr. Recurrent Instance Segmentation. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2016.

[186] O. Ronneberger, P.Fischer, and T. Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Proc. of the Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015.

[187] M. Rußwurm and M. Körner. Self-supervised contrastive learning for crop type mapping with sentinel-2 time series. In *Proc. of the Europ. Conf. on Computer Vision Workshops*, 2020.

[188] H.M. Sahin, T. Miftahushudur, B. Grieve, and H. Yin. Segmentation of weeds and crops using multispectral imaging and crf-enhanced u-net. *Computers and Electronics in Agriculture*, 211:107956, 2023.

[189] X. Saining, G. Jiatato, G. Demi, Q. Charles R., G. Leonidas, and L. Or. PointContrast: Unsupervised Pre-Training for 3D Point Cloud Understanding. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2020.

[190] M.S. Sajjadi, O. Bachem, M. Lucic, O. Bousquet, and S. Gelly. Assessing generative models via precision and recall. *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 31, 2018.

[191] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.C. Chen. MobileNetV2: Inverted residuals and linear bottlenecks. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[192] M.A. Saqib, M. Aqib, M.N. Tahir, and Y. Hafeez. Towards deep learning based smart farming for intelligent weeds management in crops. *Frontiers in Plant Science*, 14, 2023.

[193] D. Schunck, F. Magistri, R. Rosu, A. Cornelißen, N. Chebrolu, S. Paulus, J. Léon, S. Behnke, C. Stachniss, H. Kuhlmann, and L. Klingbeil. Pheno4D: A spatio-temporal dataset of maize and tomato plant point clouds for phenotyping and advanced plant analysis . *PLOS One*, 16(8):1–18, 2021.

[194] M. Sensoy, L. Kaplan, and M. Kandemir. Evidential deep learning to quantify classification uncertainty. In *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 2018.

[195] S. Shao, Y. Bai, Y. Wang, B. Liu, and B. Liu. Collaborative consortium of foundation models for open-world few-shot learning. *AAAI Conference on Artificial Intelligence*, 38(5), 2024.

[196] M.A. Shirzi and M.R. Kermani. Adaptive feature-based plant recognition. *IEEE Trans. on AgriFood Electronics*, 2(2):335–346, 2024.

[197] J. Sietsma and R.J. Dow. Creating artificial neural networks that generalize. *Neural Networks*, 4(1):67–79, 1991.

[198] B. Singh, S. Kumar, A. Elangovan, D. Vasht, S. Arya, N.T. Duc, P. Swami, G.S. Pawar, D. Raju, H. Krishna, et al. Phenomics based prediction of plant biomass and leaf area in wheat using machine learning approaches. *Frontiers in Plant Science*, 14:1214801, 2023.

[199] R.K. Srivastava, K. Greff, and J. Schmidhuber. Training very deep networks. In *Proc. of the Conf. Neural Information Processing Systems (NIPS)*, 2015.

[200] M.P.J. Tabe-Ojong, J.C. Lokossou, B. Gebrekidan, and H.D. Affognon. Adoption of climate-resilient groundnut varieties increases agricultural production, consumption, and smallholder commercialization in west africa. *Nature Communications*, 14(1):5175, 2023.

[201] B.G. Tamang, Y. Zhang, M.A. Zambrano, and E.A. Ainsworth. Anatomical determinants of gas exchange and hydraulics vary with leaf shape in soybean. *Annals of Botany*, 131(6):909–920, 2022.

[202] A. Tamborrino, N. Pellicanò, B. Pannier, P. Voitot, and L. Naudin. Pretraining is (almost) all you need: An application to commonsense reasoning. In *Proc. of the Association for Computational Linguistics*, pages 3878–3887, 2020.

[203] Y. Tang, D. Yang, W. Li, H.R. Roth, B. Landman, D. Xu, V. Nath, and A. Hatamizadeh. Self-supervised pre-training of swin transformers for 3d

medical image analysis. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[204] J.R. Teasdale and D.W. Shirley. Influence of herbicide application timing on corn production in a hairy vetch cover crop. *Journal of Production Agriculture*, 11(1):121–125, 1998.

[205] H. Thomas, C. Qi, J. Deschaud, B. Marcotegui, F. Goulette, and L. Guibas. KPConv: Flexible and Deformable Convolution for Point Clouds. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2019.

[206] F. Tomita, M. Yachida, and S. Tsuji. Detection of homogeneous regions by structural analysis. In *Proc. of the Intl. Conf. on Artificial Intelligence (IJCAI)*, 1973.

[207] B. Triggs, P.F. McLauchlan, R.I. Hartley, and A.W. Fitzgibbon. Bundle adjustment - a modern synthesis. In *Proc. of the Intl. Workshop on Vision Algorithms: Theory and Practice*, 1999.

[208] E. Tuba, R. Jovanovic, and M. Tuba. Plant diseases detection based on color features and kapur's method. *Trans. on Information Science and Applications*, 14:31–39, 2017.

[209] J. Ubbens, M. Cieslak, P. Prusinkiewicz, and I. Stavness. The use of plant models in deep learning: an application to leaf counting in rosette plants. *Plant Methods*, 14:1–10, 2018.

[210] V. Udandarao, A. Prabhu, A. Ghosh, Y. Sharma, P. Torr, A. Bibi, S. Albanie, and M. Bethge. No" zero-shot" without exponential data: Pretraining concept frequency determines multimodal model performance. In *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 2024.

[211] United Nations. Department of Economic and Social Affairs. Population Division. World Population Prospects 2022: Methodology of the United Nations Population Estimates and Projections. 2022.

[212] S. Van der Walt, J.L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J.D. Warner, N. Yager, E. Gouillart, and T. Yu. scikit-image: image processing in python. *PeerJ*, 2:e453, 2014.

[213] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, and I. Polosukhin. Attention Is All You Need. In *Proc. of the Conf. Neural Information Processing Systems (NIPS)*, 2017.

[214] O. Vysotska, H. Kuhlmann, and C. Stachniss. UAVs Towards Sustainable Crop Production. In *Workshop at Robotics: Science and Systems*, 2019.

[215] G. Wang, K. Wang, G. Wang, P. Torr, and L. Lin. Solving Inefficiency of Self-supervised Representation Learning. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2021.

[216] L. Wang, L. Zheng, and M. Wang. 3D Point Cloud Instance Segmentation of Lettuce Based on PartNet. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[217] R. Wang, M. Schwörer, and D. Cremers. Stereo DSO: Large-Scale Direct Sparse Visual Odometry with Stereo Cameras. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2017.

[218] S. Wang and R.M. Haralick. Automatic multithreshold selection. *Computer Vision, Graphics, and Image Processing*, 25(1):46–67, 1984.

[219] C. Wanh, Y. Xia, L. Xia, Q. Wang, and L. Gu. Dual discriminator gan-based synthetic crop disease image generation for precise crop disease identification. *Plant Methods*, 21:46, 2025.

[220] M. Watt, F. Fiorani, B. Usadel, U. Rascher, O. Muller, and U. Schurr. Phenotyping: New windows into the plant for breeders. *Annual Review of Plant Biology*, 71:689–712, 2020.

[221] S.M. Weraduwage, J. Chen, F.C. Anozie, A. Morales, S.E. Weise, and T.D. Sharkey. The relationship between leaf area growth and biomass accumulation in Arabidopsis thaliana. *Frontiers in Plant Science*, 6:167, 2015.

[222] J. Weyler, F. Magistri, P. Seitz, J. Behley, and C. Stachniss. In-Field Phenotyping Based on Crop Leaf and Plant Instance Segmentation. In *Proc. of the IEEE Winter Conf. on Applications of Computer Vision (WACV)*, 2022.

[223] J. Weyler, A. Milioto, T. Falck, J. Behley, and C. Stachniss. Joint Plant Instance Detection and Leaf Count Estimation for In-Field Plant Phenotyping. *IEEE Robotics and Automation Letters (RA-L)*, 6(2):3599–3606, 2021.

[224] J. Weyler, F. Magistri, E. Marks, Y.L. Chong, M. Sodano, G. Roggiolani, N. Chebrolu, C. Stachniss, and J. Behley. Phenobench: A large dataset and benchmarks for semantic image interpretation in the agricultural domain. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, 46(12):9583–9594, 2024.

[225] J. Weyler, J. Quakernack, P. Lottes, J. Behley, and C. Stachniss. Joint plant and leaf instance segmentation on field-scale uav imagery. *IEEE Robotics and Automation Letters (RA-L)*, 7(2):3787–3794, 2022.

[226] W. Winterhalter, F.V. Fleckenstein, C. Dornhege, and W. Burgard. Crop Row Detection on Tiny Plants With the Pattern Hough Transform. *IEEE Robotics and Automation Letters (RA-L)*, 3(4):3394–3401, 2018.

[227] D.M. Woebbecke, G.E. Meyer, K.V. Bargen, and D.A. Mortensen. Color indices for weed identification under various soil, residue, and lighting conditions. *Transactions of the American Society of Agricultural and Biological Engineers (ASABE)*, 38:259–269, 1994.

[228] D.M. Woebbecke, G.E. Meyer, K. Von Bargen, and D.A. Mortensen. Color indices for weed identification under various soil, residue, and lighting conditions. *Trans. of the American Society of Agricultural Engineers*, 38(1):259–269, 1995.

[229] B. Xiao, H. Wu, W. Xu, X. Dai, H. Hu, Y. Lu, M. Zeng, C. Liu, and L. Yuan. Florence-2: Advancing a unified representation for a variety of vision tasks. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[230] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J.M. Alvarez, and P. Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. In *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 2021.

[231] R. Yamada, H. Kataoka, N. Chiba, Y. Domae, and T. Ogata. Point Cloud Pre-Training With Natural 3D Structures. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[232] J. Yang, J. Liu, N. Xu, and J. Huang. Tvt: Transferable vision transformer for unsupervised domain adaptation. In *Proc. of the IEEE Winter Conf. on Applications of Computer Vision (WACV)*, 2023.

[233] S. Yang, L. Zheng, T. Wu, S. Sun, M. Zhang, M. Li, and M. Wang. High-throughput soybean pods high-quality segmentation and seed-per-pod estimation for soybean plant breeding. *Engineering Applications of Artificial Intelligence*, 129:107580, 2024.

[234] W. Yang, H. Feng, X. Zhang, J. Zhang, J.H. Doonan, W.D. Batchelor, L. Xiong, and J. Yan. Crop phenomics and high-throughput phenotyping: past decades, current challenges, and future perspectives. *Molecular Plant*, 13(2):187–214, 2020.

[235] S. Yanowitz and A. Bruckstein. A new method for image segmentation. *Computer Vision, Graphics, and Image Processing*, 46(1):82–95, 1989.

[236] S. Yoshimoto. A study on artificial neural network generalization capability. In *Proc. of the Intl. Joint Conf. on Neural Networks (IJCNN)*, 1990.

[237] B. Zadrozny and C. Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *Proc. of Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 694–699, 2002.

[238] J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *Proc. of the Intl. Conf. on Machine Learning (ICML)*, 2021.

[239] Y. Zeng, D. Hao, G. Badgley, A. Damm, U. Rascher, Y. Ryu, J. Johnson, V. Krieger, S. Wu, H. Qiu, Y. Liu, J.A. Berry, and M. Chen. Estimating near-infrared reflectance of vegetation from hyperspectral data. *Remote Sensing of Environment*, 267:112723, 2021.

[240] R. Zenkl, R. Timofte, N. Kirchgessner, L. Roth, A. Hund, L. Van Gool, A. Walter, and H. Aasen. Outdoor plant segmentation with deep learning for high-throughput field phenotyping on a diverse wheat dataset. *Frontiers in Plant Science*, 12, 2022.

[241] H. Zhang, M. Cisse, Y.N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. In *Proc. of the Intl. Conf. on Learning Representations (ICLR)*, 2018.

[242] J. Zhang, X. Yu, A. Li, P. Song, B. Liu, and Y. Dai. Weakly-supervised salient object detection via scribble annotations. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[243] Z. Zhang, R. Girdhar, A. Joulin, and I. Misra. Self-Supervised Pretraining of 3D Features on Any Point-Cloud. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2021.

[244] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang. Random erasing data augmentation. In *Proc. of the Conf. on Advancements of Artificial Intelligence (AAAI)*, 2020.

[245] Z.H. Zhou. A brief introduction to weakly supervised learning. *National Science Review*, 5(1):44–53, 2018.

List of Figures

List of Tables